Robust 2D Model-Based Object Recognition

Todd Anthony Cass

Massachusetts Institute Of Technology
Artificial Intelligence Laboratory
May 1988

Submitted as the thesis in partial fulfillment of the requirements for the degree of Master of Science in Electrical Engineering and Computer Science under the title "Parallel Computation in Model-Based Recognition" May, 1988.

Abstract

Techniques, suitable for parallel implementation, for robust two-dimensional model-based object recognition are studied. Object models and input sensory data are represented as local geometric features. The thesis concentrates on the hypothesis of model feature to image feature matchings, and transformations between the matched features. Bounds on the error in image feature extraction are assumed, allowing the formulation of constraints on possible feature matchings and transformations. The hypothesis of feature matchings is then formulated as a search through the space of possible transformations, based on these constraints. *Transformation sampling* is introduced as a simple, robust, and highly parallel method of searching this space to hypothesize feature matchings. The time complexity and the processor requirements of this procedure are polynomial in the number of model and image features. A key feature of the approach is that error in image feature measurement is explicitly accounted for in the hypothesis process. Transformation sampling carefully samples the space of model to image feature transformations to determine transformations consistent with the model, observed data, and known feature measurement error. Three approaches to matching hypothesis by transformation sampling are developed formally: *critical point* sampling, *uniform* sampling, and *probabilistic* sampling. The results of simulations and experiments on real images based on a Connection Machine parallel supercomputer implementation are presented.

Thesis Supervisor: W. Eric L. Grimson

Acknowledgments

I thank my advisor, Eric Grimson, for first offering me the opportunity to explore the field of machine vision, and for his patient and easy-going style. There are many people with whom I have had many fun and interesting discussions about my work, machine vision in general, and beyond; and from whom I have learned much. Among them are Guy Blelloch, David Clemens, Ed Gamble, Walter Gillett, Jim Little, and Brian Subirana. I owe special thanks to my office mate, David Jacobs, who was always enthusiastic and interested in talking about new ideas. Many of the ideas in this thesis evolved from discussions with him. Finally I thank my family, my parents Ron and Marilyn and brothers and sister Terry, Tracy, Tim and Tom for their continual support in whatever endeavor I have chosen to pursue.

This report describes research done at the Artificial Intelligence Laboratory of the Massachusetts Institute of Technology, and was funded in part by an Office of Naval Research University Initiative grant under contract N00014.86K-0685, in part by the Advanced Research Projects Agency of the Department of Defense under contract N00014.85K-0124, and Army contract DACA 76-85-C.0010.

Chapter 1

Introduction

I am interested in enabling a computer or robot to utilize visual sensory data to reason about, and interact with, its environment. A simple example of this is the use of robots in the automation of manufacturing and assembly, where the use of visual feedback would allow greater flexibility in the classes of operations and the complexity of manipulations performed, by providing the robot with the capability of identifying and locating objects of interest. It seems clear that this is in general an extremely complex task; many years of research have only begun to offer understanding of the general problem of interpreting the environment through visual sensory input. There has, however, been much success in visual sensing in more restricted domains. An example of this is the area of model-based object recognition. In this domain, the goal is to identify objects in the environment using sensory data, assuming that the information relevant to the identification task is captured by a *model* of the object of interest. This thesis is focused on the development of a model based recognition technique that is effective and easily implemented on a multi-processor computer.

1.1 The Problem

Simply stated, the object recognition problem is as follows. Given as inputs a model of an object and the output of sensors in the environment of interest, the problem is to determine if the modeled object is in the environment, and if so its location. Among the many issues to be addressed are the fact that the object of interest is not isolated, but may be partially occluded by unknown objects, and that the sensor output is subject to errors. These issues are described more precisely in Chapter 2.

1.2 Goals

The area of model-based recognition has received considerable attention in recent years, and several techniques have proven effective in restricted task domains[12][5][16][8] where particular assumptions are made such as the type and number of objects recognized, and the nature of the sensory data. Examples include recognition of rigid 2D objects from 2D video images, and 3D rigid objects from 3D range data. In most cases, research has concentrated on algorithmic approaches intended for single processor computers, although some of the existing systems are likely to have efficient parallel implementations.

One of the primary goals of this thesis is to explore the issues involved with model-based recognition and develop a model-based recognition technique suitable for implementation on a multi-processor parallel computer. To be specific, the domain considered in this work is that of recognition of two-dimensional objects from two-dimensional sensory data. The primary goal is to develop a recognition technique satisfying the following general criteria:

- Recognize 2D objects from 2D sensory input
- Robust under the effects of sensing errors
- Robust under the effects of object occlusion
- Robust in the presence of unknown objects
- Efficiently implemented on a practical parallel machine.

1.3 Motivation

Model-based object recognition itself is obviously useful in tasks already common to robots such as various aspects of manufacturing and assembly. A more flexible and useful robot can be achieved by enabling it to interact dynamically with its environment using all available sensing methods, including visual sensing. But to be generally useful, these recognition systems must be able to operate at speeds comparable to the rate at which the robot operates. Because of the complexity of the task, this is not in general possible on a standard single processor computer. This motivates the study of recognition methods suitable for efficient and practical parallel implementation: parallel execution can offer substantial speedup over serial execution, depending on the extent to which an algorithm can be developed to fully exploit the available processing power.

I believe the path to developing a parallel procedure for recognition starts with understanding the basic operations from which a recognition procedure is built. From this we can determine the requirements of a recognition system in terms of general purpose parallel hardware and parallel algorithms. Another important aim of developing an understanding of the basic components of robot recognition, and finding simple, highly parallel techniques is that it will then be possible to build special purpose parallel hardware to execute the procedures extremely quickly. While actual hardware implementation of parallel recognition may still be out of reach, existing parallel computers provide excellent tools for research in this area.

Why study 2D recognition? A considerable amount of work has already been done in the area of recognizing 2D objects from 2D visual images[12]. However, few people have demonstrated working parallel systems. Ultimately we want to be able to achieve recognition of 3D objects from 3D and 2D data very fast, as humans can, for applications in robotics. There are three main reasons why it is interesting to study 2D recognition. First, 2D recognition is is interesting and useful in its own right, and the fast computation of 2D recognition would be useful. Second, 2D recognition is a good starting place for developing needed insights into the issues involved with parallel computation in general, and in particular parallel computation of object recognition. Lastly, with care in the direction taken, studying 2D parallel recognition can be a useful step toward understanding 3D parallel recognition. In fact, while I have studied so called *2D from 2D* recognition, some of the techniques developed here appear promising for extensions to *3D from 3D* recognition.

1.4 Contributions

This work provides an elegant and simple characterization of a commonly used formulation of 2D recognition. Based on this characterization, a provably correct algorithm for recognition is developed under a particular definition of recognition and assumptions on the nature of measurement errors. Two other sub-optimal, but highly parallel, algorithms are developed and studied. The algorithms are extremely simple in terms of the control structure and the types of computations that are required, making them easily implemented in parallel, and candidates for a special hardware implementation of recognition.

Chapter 2

Background

Model-based recognition seeks to exploit knowledge of an object, in the form of a model, to recognize it. In general, a model of an object captures information sufficient for recognizing an instance of the object and for distinguishing it from other objects in the domain. There are a variety of attributes that can be considered such as color, texture, and shape. Many techniques are based solely on object shape, and this is the approach taken here. Chapter 6 provides a brief overview of model-based recognition techniques, but the following brief characterization provides some context for the approach to recognition taken here.

The types of techniques can be roughly divided into three categories: those in which the model does not maintain a representation of the spatial structure of the object, those that maintain a qualitative representation of the spatial structure of an object, and those that maintain an explicit quantitative geometrical representation of the spatial structure of an object. Among the last group are techniques that require that the modeled object is rigid and unchanging in structure. The system described here relies on this restriction.

This chapter serves to introduce the terminology used throughout this document, and the concepts central to model-based object recognition.

2.1 Models, Features, and Image Data

The inputs to a model-based recognition system consist of a set of models, and sensory data derived from the environment. The recognition procedure attempts to interpret and explain, if possible, the sensory data in terms of one or more of the models. To facilitate this process, the model and the sensory data must be represented in a compatible manner.

For shape-based recognition of 2D objects, the boundary contour of an object captures

all the necessary information, and thus logically might form the basis for the object representation. In this work, the model forms a representation of the boundary contour in terms of contour *features*. These are *local features* in that they each represent a small portion of the boundary contour. Very few restrictions are placed on what constitutes a feature, however the two requirements are:

- A feature has a definite orientation
- A feature has a definite position

The two types of contour features used in this work are *point features* consisting the position of a point on the contour and the direction of the contour normal at the point, and *line segments* consisting of a straight line approximation of a contour segment. To facilitate recognition, both the model and the sensory data are represented in exactly the same manner, as a set of contour features representing the boundary contours of the model and the sensed objects, respectively.

The sensory data are commonly in the form of video images, sonar or laser range data, or tactile data. *Feature extraction* is the process of representing the sensory data in terms of features for recognition. In the case of a CCD video image, contour features, and 2D objects as in this work, the feature extraction process is composed of an edge detection phase to extract the boundary contours, followed by a feature building stage forming features out of segments of the boundary contour. For convenience, in most of this document I assume that the sensory data consist of a discrete CCD brightness image, and thus the input data are in the form of an image brightness array.

2.1.1 Uncertainty, Occlusion, and Spurious Data

Although the features representing the sensory data are defined in terms of a definite position, and orientation, I assume that noise in the input sensor, spatial distortions in the sensing device, and approximations in feature extraction result in *uncertainty* in the position and orientation of each image feature. This means that the measured position and orientation of a feature extracted from the sensed data may differ from its actual position and orientation in the environment. The mechanisms causing uncertainty are beyond the scope of this work, however I use two simple models of uncertainty. In the first, uncertainty is modeled as completely unknown but bounded in magnitude by a known constant. In the second, the uncertainty is characterized by a probability distribution on the range of possible deviations from the correct pose. This is discussed further in chapter 3.

In addition to sensing errors there are other factors complicating the sensing of objects in the environment. First, the data from part of an object may not be available due to *occlusion* by other objects or other irregularities in sensing. Occlusion prevents the features corresponding to the obscured part of the object from being extracted from the input data, or results in image features which are fragments of the model feature to which they correspond. Second, the object of interest may not be alone in the scene, rather there may be several unknown objects in the scene. These *spurious* data may be locally indistinguishable from the correct data, and serve only to complicate the recognition process.

2.2 Recognition

The simplest definition of object recognition is that it consists of determining the presence and identity of an object in the sensed environment. For robotics applications, it is natural to also require the determination of the position and orientation, or *pose*, of the object in the environment. In general, any one of several known objects could appear in the scene, and to be generally useful a system must be able to recognize all instances of any one of the possible known objects in the image. This problem of recognizing an object out of a large possible library is very difficult in itself, and beyond the scope of this thesis. Chapter 7 briefly discusses these issues. This thesis focuses on the recognition of objects when it is known which object to expect in the image. In this thesis, recognition will basically consist of:

- Determining whether the object is in the scene.
- Determining the *pose* of the object in the environment.

The fundamental assumption of model based vision, and in particular recognition of rigid objects, is that there exists a correspondence between the model features, and those image features due to the object. The correct feature correspondence can be sufficient to determine the identity of the object, and in the case of rigid objects the correct feature correspondence can be used to derive the object's pose. Let $\{m_i\}$ describe the set of model features, and $\{d_j\}$ describe the set of image features, and let $m = |\{m_i\}|$ and $n = |\{d_j\}|$. A particular feature match is denoted by $< m_i, d_j >$; and $S = \{< m_i, d_j >\} = \{m_i\} \times \{d_j\}$ denotes the set of all possible feature matches. We say that a subset $\mathbf{M} \subseteq S$ is a *matching* between model features and data features. A *correct* matching is a matching in which each model feature is correctly matched to its corresponding image feature.

The pose of the model of an object, and the pose of the image of the object can be related by a transformation. Assume the ideal case where the image features have been extracted without error. For 2D objects there exists a *transformation* $T = (\phi, u, v)$ consisting of a rotation ϕ followed by a translation $\mathbf{t} = [u \quad v]^T$ on each of the model features which exactly aligns the model with its image. Because these are 2D vectors, it is convenient to represent them as complex numbers. Let the complex numbers $\mathbf{p}_m$ and $\mathbf{p}_d$ represent the position of a model feature and an image feature, respectively. The transformation between the pose of the model feature and the pose of the image feature is given by

$$\mathbf{t} = \mathbf{p}_d - e^{i\phi}\mathbf{p}_m$$

where $\phi = \theta_d - \theta_m$ is the difference in orientation of the image and model feature respectively, and multiplication by $e^{i\phi}$ rotates θ_m through an angle ϕ. Unless specifically stated otherwise, for convenience all following discussion is of 2D objects and 2D sensory data. Because the pose of an object in the environment is related to the pose of the image of the object in sensor coordinates by a known transformation, the determination of the model object to image object transformation relates the known pose of the model to the pose of the object in the environment.

The concept of *transformation parameter space* provides a convenient tool to deal with transformations, and plays a crucial role in the formal development of the recognition techniques described herein. Each transformation can be represented as a point (ϕ, u, v) in the the transformation parameter space $\mathbf{TPS} = \Re^2 \times SO_2$ where SO_2 is the 2D rotation group, ϕ the rotation and u and v the translation in the x and y directions respectively. When the image features are of unknown scale, the scale factor s becomes a fourth parameter of the transformation, and $\mathbf{TPS} = \Re^3 \times SO_2$. Initially I will assume scale is known.

2.2.1 Matchings and Transformations

We can now view the recognition problem as the problem of determining a correct feature matching and a transformation between the model pose and the image object pose. As we shall see, when the models and objects are rigid, the problem of constructing a correct feature match and determining a transformation complement one another.

To illustrate the interaction between constructing a matching and determining a transformation, it is instructive to consider a somewhat ideal case. Assume that scale is known and that there is no uncertainty in the pose of an image feature; each image feature corresponds precisely to the object that gave rise to it. In this case a single correct feature

match is sufficient to derive the transformation from the model pose to the image pose. Constructing a matching and determining a transformation are intimately connected. Each feature match implies a transformation and a matching.

When image feature pose uncertainty is included in the analysis, a correct feature match no longer necessarily implies a transformation equating the poses of correctly matched features, and determining a matching becomes much more difficult. Altogether there are three main factors which make finding a matching difficult:

- Due to object occlusion in the scene, some model features may not correspond to any image feature. These are *missing* or *occluded* model features.
- Due to *spurious* image data from unknown objects in the scene, some image features will not correspond to any model feature.
- There is *uncertainty* as to the exact pose of an image feature.

In the ideal case just described, there were no more than mn matchings that needed to be considered, each one being verified in at most mn steps by transforming each of the m model features according to each of the mn possible transformations, and then comparing it to each of the n image features to find which one it aligns with exactly, for a total complexity of $m(mn)n = m^2n^2$.[1] When the measured pose of an image feature is allowed to vary from its actual pose, however, it is not as easy to verify membership in a matching since a match could be part of the correct matching yet not align exactly at the correct transformation. Thus, the simple method of constructing a matching may not work. There are $(n+1)^m$ possible matchings in which each image feature appears once[16], so one possible approach is to simply consider each of the $(n+1)^m$ possible matchings, clearly intractable. Even if we could, however, we would still need some criteria for selecting the best match.

2.2.2 Formalizing Recognition

To make the recognition procedure more formal we require some means of evaluating matchings to determine the optimal ones. In particular, this is important when image feature pose uncertainty is considered because in this case no single transformation is clearly correct. Let $F(\mathbf{M}, T)$ be a metric, $F : \{(\mathbf{M}, T)\} \longrightarrow \Re$, where $\mathbf{M} = \{< m_i, d_j >\}$ is a feature matching and $T = (\phi, u, v)$ is a transformation. $F(\mathbf{M}, T)$ provides a measure of the quality

[1] In practice one might improve upon this bound by using a hashing technique to find the image feature corresponding to a particular model feature.

of fit between the transformed model and the matched image features. The quality of a matching $\mathbf{M}$ can be evaluated by finding the optimal value of $F(\mathbf{M}, T)$ over all transformations T. Let $f(< m, d >, T)$ be a metric on the difference in the pose of the features for a particular match. One possible construction of F is as some function of $f(< m, d >, T)$ over all matches in the matching, in which case $F(\mathbf{M}, T)$ takes account of how well individual feature matches match one another after transformation.

In this work, recognition will be formally structured according to the common hypothesis generation and verification paradigm. In the hypothesis generation phase, hypotheses of matchings $\mathbf{M}$ and transformations T are constructed which optimize the metric $F(\mathbf{M}, T)$. These hypotheses are then verified by comparing the matched, transformed model and image features sets, or by using lower-level representations of the original input image and modeled object. The results of recognition are those optimal hypotheses which pass the verification process.

2.3 The Bounded Uncertainty Assumption

The previous sections discussed the construction and evaluation of matches. Spurious and occluded data as well as image feature pose uncertainty conspire to make determining an optimal matching difficult. This section introduces the idea of constructing an approximation to an optimal matching. The basic idea is to allow features to be matched after transformation only if the distance between their resulting poses, measured by the metric f, is below some bound.

Let D denote an upper bound on the magnitude of the position uncertainty of an image feature, and Θ denotes an upper bound on the orientation uncertainty of an image feature. To apply the bounded uncertainty approximation we assume that the position error $\delta\vec{p}$, and the orientation error θ_e, obey $|\delta\vec{p}| \leq D$ and $|\theta_e| \leq \Theta$ respectively. In this case we do not allow two features to match unless their position and orientation are within these bounds of each other. Many investigators have explicitly or implicitly applied this approximation[16][4][8][24], for example this follows closely the extremely effective constraints applied by Grimson and Lozano-Pérez[16].

By assuming uncertainty is bounded, the number of possible matchings is greatly reduced. In fact, any transformation defines a matching. This idea, along with metrics for evaluating matchings will form the basis of the recognition techniques to be developed in the next chapter.

2.4 Summary

The known objects are represented by models consisting of a set of local contour features with position and orientation. The image data are abstracted, and the objects in the image represented in the same way, as a set of contour features. There is uncertainty in the measurement of the position and orientation of a feature extracted from the image, but the magnitude of the uncertainty is bounded.

Generally, recognition consists of determining if there is an instance of the modeled object in the input image. More formally, recognition is defined as hypothesizing a matching M and transformation T optimizing the metric $F(\mathrm{M}, T)$. These hypotheses are verified by some comparison between the model and some representation of the image data.

Chapter 3

Recognition as Searching Parameter Space

Chapter 2 defined model-based recognition for rigid models very generally. Recognition consists of the hypothesis of matchings M and transformation T, followed by the verification of these hypotheses. Hypothesis and verification provide an explanation for observed image features in terms of the model. This chapter outlines how hypotheses are constructed by searching the set of possible matchings M and transformations T for optima in the function $F(\mathbf{M}, T)$.

3.1 Recognition Strategy

Feasible Matchings

The notion of a feasible matching forms the basis for the strategy for constructing hypotheses for M and T. Under the assumption of bounded image feature pose uncertainty, a feature match is *feasible* if for some transformation, T, on the model feature, the difference in the position and orientation of the two features is within the uncertainty bounds. A matching is feasible if there exits some T at which all its component matchings are feasible. The approach taken here is to divide the hypothesis stage into two steps. First, *hypothesis generation* constructs a set of feasible matchings. Second, *Hypothesis refinement* determines for each such feasible match, the associated transformation T which optimizes $F(\mathbf{M}, T)$. The verification stage then determines if the hypothesis is correct by utilizing any available image and model data that is appropriate.

The key to the hypothesis step is the construction of feasible matchings accounting for significant portions of the image features. In previous work, considerable computational effort was placed on this process[16][8][1] The approach to constructing feasible matchings taken here is to search **TPS** for translations T associated with large feasible matchings. The idea is that any transformation T implies a feasible matching consisting of all matches feasible at T. As we shall see, a particular matching may be feasible over a whole range of transformations T, corresponding to a region in **TPS**. Hypothesis generation will be formulated as a search for these regions. So, to facilitate this strategy, and structure the search for optima of $F(\mathbf{M}, T)$ as a search over all T in the transformation parameter space **TPS**, we simplify $F(\mathbf{M}, T)$ as follows. Define

$$F(\mathbf{M}, T) = F'(\mathbf{M}(T)) = F''(T)$$

and dropping the primes for convenience, define

$$F(T) = \sum_{<m_i, d_j> \in \mathbf{M}} f(< m_i, d_j >, T)$$

where $\mathbf{M}$ consists of matches feasible at T and $f(< m_i, d_j >, T) = c_{ij} > 0$ for T within the range of feasible transformations of match $< m_i, d_j >$, and 0 outside it; where c_{ij} is a constant depending on the match $< m_i, d_j >$.

In this special case, $F(T)$ is a piecewise constant function over **TPS** changing values only at the boundary of regions of feasible transformations for particular matches. $F(T)$ does not depend on $\mathbf{M}$ because a transformation T implicitly defines a matching consisting of all feasible matches at T.

This particular definition of $F(T)$ facilitates the hypothesis generation step: finding feasible matchings. By appropriate definition of the constants c_{ij}, maxima of $F(T)$ are likely to correspond to largely correct matchings. An example is to define c_{ij} to be the length of the image feature for the match $< m_i, d_j >$. In this case a maximum of $F(T)$ defines a matching for which there exist T where all matched features satisfy the uncertainty constraints, and which accounts for a large fraction of the image features in terms of the model.

The hypothesis generation step can now be viewed as a search of **TPS** for one of these piecewise constant regions with an optimal value of $F(T)$. Because with this definition $F(T)$ is piecewise constant, there is a whole range of T which imply the same matching $\mathbf{M}$.

[1]Note that feasible is defined in a slightly different sense in [16]. Here feasible means there exists a T where all matchings satisfy the constraints on uncertainty.

The purpose of hypothesis refinement step is to refine the hypotheses by utilizing a different definition of $F(T)$ and determining which T are optimal for the matching **M**. Because the hypothesis of matchings is the most difficult step, the primary emphasis in this thesis is placed on hypothesis generation, the hypothesis of matchings. Section 3.6 discusses some simple methods of hypothesis refinement.

3.2 The Structure of Parameter Space

The first step of recognition, hypothesis generation, is viewed as searching **TPS** for optimal values of $F(T)$, using the special definition of the previous section. To facilitate this it is necessary to understand the structure of the transformation parameter space.

Define a *feature-match* to be the pairing of a model feature with an image feature. The range of rotations which will bring the orientation of the model feature to within Θ of the orientation of the image feature, and the range of translations, after each of these rotations, which bring the position of the model feature to within D of the image feature correspond to a region in transform parameter space of feasible transformations called a *match-region.*

Assume the match-regions for all possible feature matches are constructed. As mentioned earlier, the boundaries of these match-regions divide **TPS** into irregularly shaped volumes, each bounded by the bounding surfaces of one or more match-regions, or the curves of intersection of two or more match regions. I call these regions of intersection of match-regions *intersection-volumes.* By the assumptions on the character of the pose uncertainty, the *correct* transformation is included within the match-region of each correctly matched image and model feature. Thus, one of the intersection-volumes contains the correct transformation, and represents a range of transformations consistent with all correct feature matches. This is the intersection-volume for which we are searching. By appropriate definition of $F(T)$, this region will have an optimal value of $F(T)$. Call an intersection volume with an optimal value of $F(T)$ an *optimal intersection-volume.*

3.3 Critical Point Sampling

We now have the general goal of hypothesis generation formulated as finding optimal intersection volumes. As introduced above, intersection volumes are formed by the intersection of match regions. Because the shape and position of the match-regions in **TPS** can be determined, one approach to finding intersection volumes would be to explicitly calculate

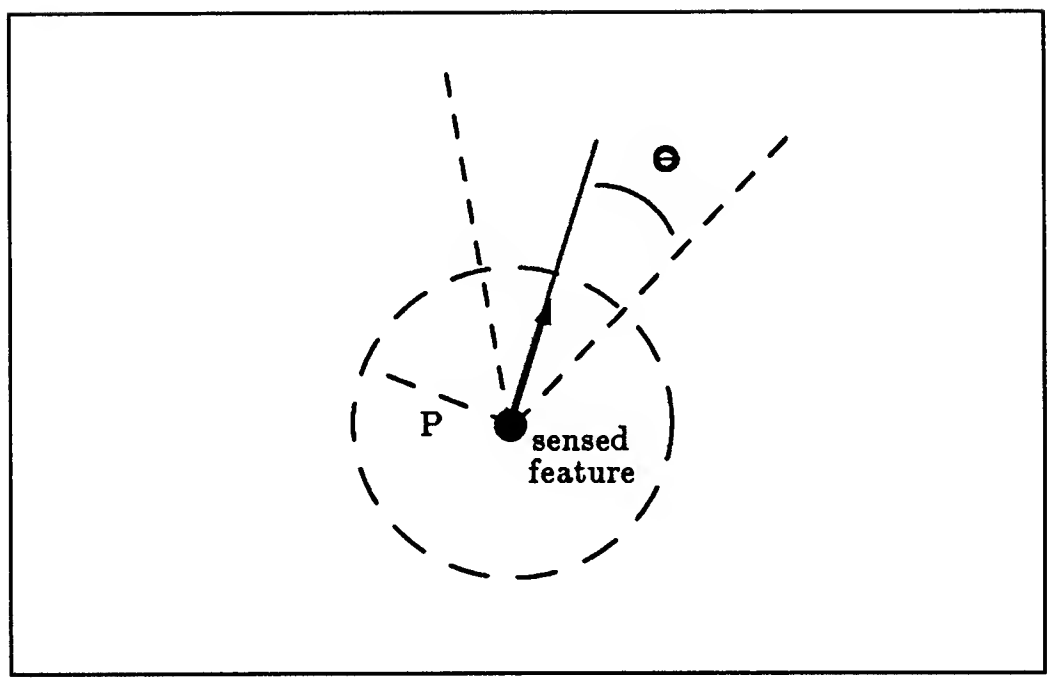

Figure 3.1: A point feature illustrated as its normal vector, and the error bounds on position a uncertainty.

their shape and position from the intersection of the associated match-regions. Although the match-regions are easily characterized, calculating the regions of their intersections is extremely difficult.

In this section I will show that it is not necessary to explicitly calculate the shape of an intersection volume to find it. The key insight is that there are certain *critical points* in **TPS** which lie on the surface of all intersection volumes. These critical points can be computed and used to locate the intersection volumes, facilitating the selection of the optimal intersection-volume, and achieving the hypothesis generation step.

For the following analysis we consider point features consisting of the location of a point on the contour of an object, and the direction of the curve normal at that point. The uncertainty in the position of an image feature is bounded by D, and the uncertainty in its orientation is bounded by Θ.

Match Regions

The difference in the position of a model feature and an image feature after rotation of the model feature by any rotation ϕ is given by the complex equation

$$\mathbf{t} = \mathbf{p}_d - e^{i\phi}\mathbf{p}_m.$$

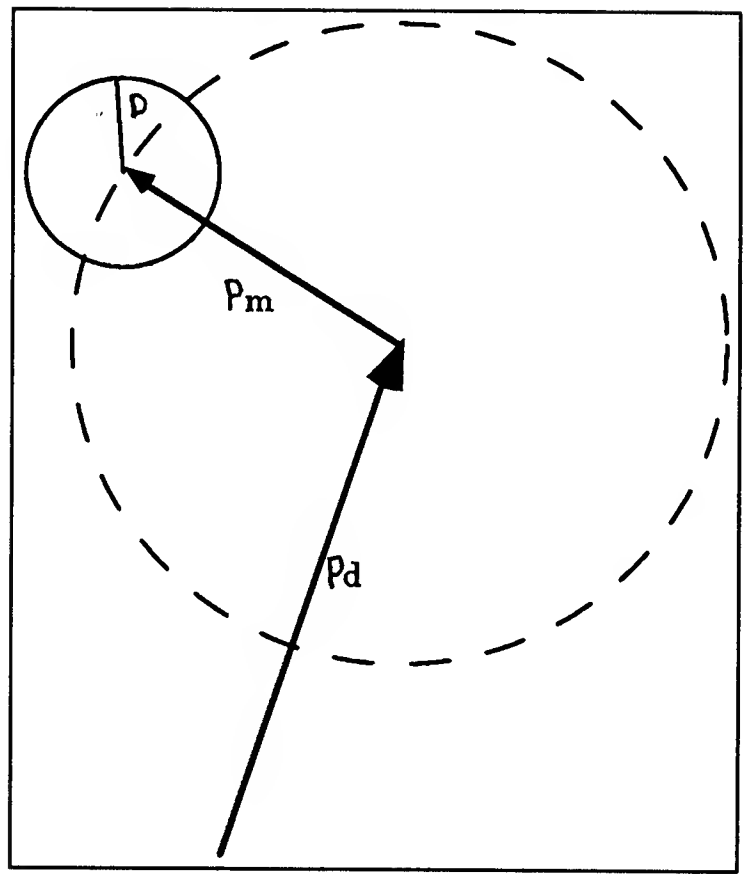

Figure 3.2: The path followed by a circle of radius D centered at tc

Let θ_d and θ_m represent the orientation of an image and model feature respectively. The *nominal* rotation aligning the orientation of the model feature with that of the image feature is given by $\phi_{dm} = \theta_d - \theta_m$. The extent of a match-region in the ϕ dimension of **TPS**, that is, the range of rotations of the model feature leaving it within Θ of the orientation of the image feature is given by the set $\phi \in [\phi_a, \phi_b]$ where $\phi_a = \phi_{dm} - \Theta$ and $\phi_b = \phi_{dm} + \Theta$.

For a particular match region, its cross section in a plane of constant ϕ in **TPS** is just a circle of radius D corresponding to the region of uncertainty in the translation aligning the positions of the features. Call this circle a *match circle*. The position, $\mathbf{t}$, of the center of a match circle in the $u - v$ plane can be thought of as a function of ϕ given by the equation above: $\mathbf{t} = \mathbf{p}_d - e^{i\phi}\mathbf{p}_m$. Thus a match circle follows a circular path of radius $|\mathbf{p}_m|$, centered at $\mathbf{p}_d$. We can easily see that the match region is described by the volume swept out by a horizontal circle of radius D following a helical path in **TPS** given by $\mathbf{t} = \mathbf{p}_d - e^{i\phi}\mathbf{p}_m$[13].

3.3.1 Critical Points in Parameter Space

I have cast hypothesis generation as a search through **TPS** for optimal intersection-volumes. While it is very difficult to explicitly compute the bounding surface of the intersection-

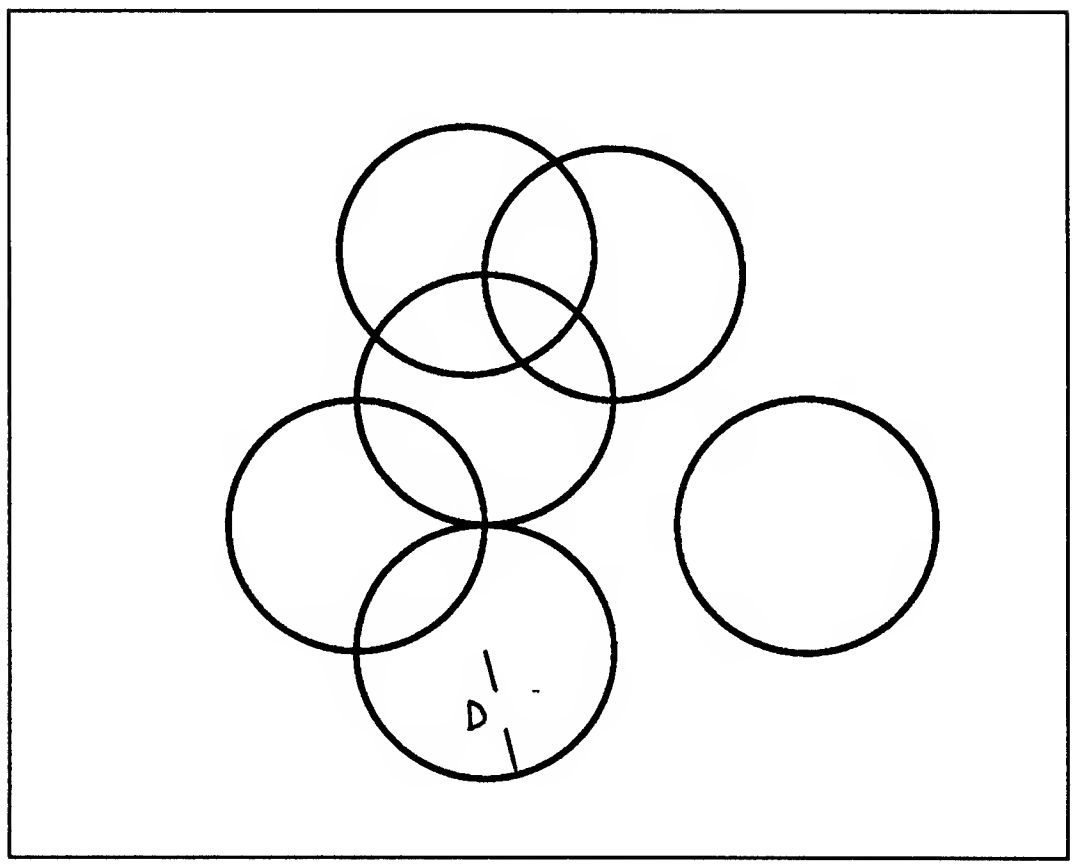

Figure 3.3: Intersection areas, slices of intersection-volumes.

volumes, there are certain computable points I call *critical points* which lie on the surface of each intersection-volume. I begin developing this idea with the following theorem which will be proved over the subsequent sections.

Theorem 1 *There is a set of points in parameter space such that at least one such point lies on the bounding surface of each intersection volume. The size of this set is polynomial in the number of match-regions.*

3.3.2 The Structure of Intersection Volumes

If we consider a slice of parameter space defined by a plane of constant ϕ, the intersection volumes have cross section consisting of the intersection in this plane of several match circles. Call the projection of such a slice of an intersection volume onto the $u - v$ plane, an *intersection-area.* See figure 3.3. Let the *value* of an intersection area be the value of $F(T)$ within the intersection area.

To characterize the structure of intersection volumes it is convenient to study the behavior of intersection areas as functions of ϕ. This is because as ϕ varies, match areas

sweep out intersection-volumes, and the boundary contours of match areas sweep out the bounding surface of intersection-volumes. An important characteristic of intersection areas is the nature of the points of intersection of the match-circles that form them. Recall that each match region exists only over a range $\phi \in [\phi_a, \phi_b]$, thus each match circle appears and disappears as ϕ passes through this range. As this process evolves with ϕ, new intersection areas appear and disappear, and as we observe the evolution of intersection areas in the (u, v) plane, there are two important patterns of behavior:

- The area of an intersection area may be non-zero over a range of ϕ including at least one of the end points ϕ_a and ϕ_b.
- The intersection area exists with non-zero area somewhere in the open interval $\phi \in (\phi_a, \phi_b)$, but not at ϕ_a or ϕ_b.

3.3.3 Intersections of Match Circles

We now examine the above two cases in more detail. As we shall see, the points where two or three match circles intersect will form the critical points sought. The following lemmas provide the formal basis for this idea, which I subsequently develop.

2-way Intersection

Lemma 1 *For a given pair of feature matches, there are at most two values of ϕ where the two match circles intersect at only one point, unless the two match circles intersect at only one point for any ϕ.*

Proof: Two match circles intersect at only one point when their centers $\mathbf{t}_1$ and $\mathbf{t}_2$ are exactly a distance $2D$ apart. In this case the centers satisfy

$$(\mathbf{t}_1 - \mathbf{t}_2)(\mathbf{t}_1^* - \mathbf{t}_2^*) = 4D^2$$

where $\mathbf{t}^*$ denotes the complex conjugate of $\mathbf{t}$. Thus we seek the roots of the equation

$$\mathbf{t}_1\mathbf{t}_1^* + \mathbf{t}_2\mathbf{t}_2^* - (\mathbf{t}_1^*\mathbf{t}_2 + \mathbf{t}_1\mathbf{t}_2^*) - 4D^2 = 0.$$

Because $\mathbf{t} = \mathbf{p}_d - e^{i\phi}\mathbf{p}_m$, and multiplying the above equation by $e^{i\phi}$ will not change its roots, inspection shows that the resulting equation is quadratic in $e^{i\phi}$ and therefore there are no more than two values of ϕ which satisfy it, or any ϕ will satisfy it when all the coefficients are zero. □

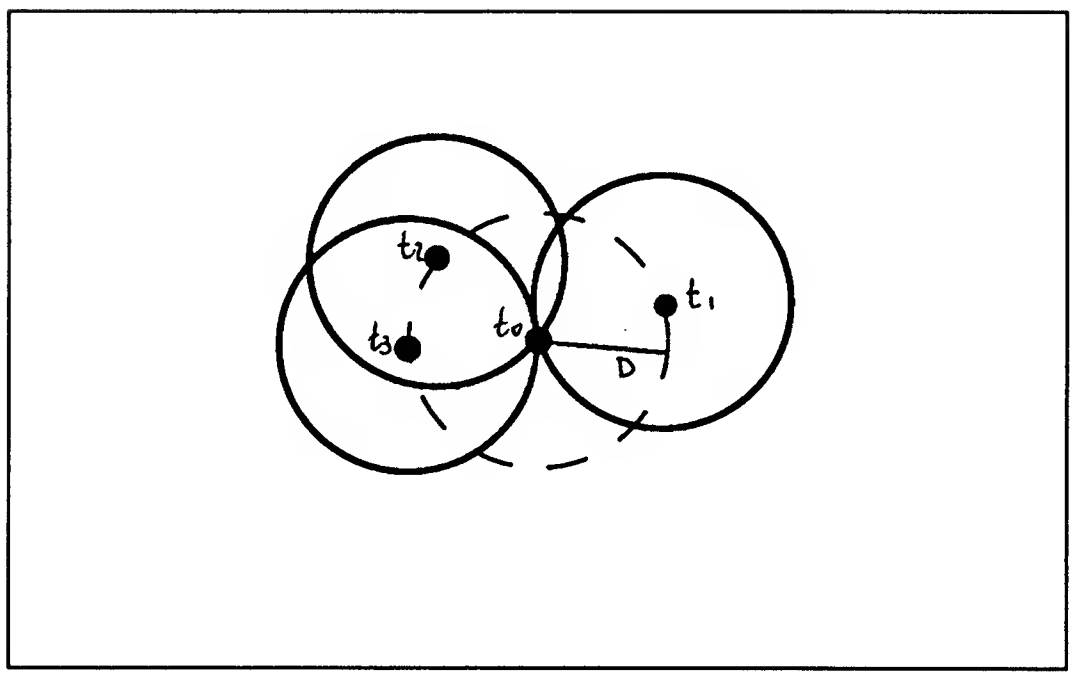

Figure 3.4: When three match circles intersect at a point, their centers fall on a particular circle of radius D

3-way Intersection

Given three match circles whose centers are described by the complex quantities $\mathbf{t}_1(\phi)$, $\mathbf{t}_2(\phi)$, and $\mathbf{t}_3(\phi)$, for which ϕ do they all intersect at a point? Note that this is equivalent to seeking those pairs $(\phi, \mathbf{t}_0)$ for which $\mathbf{t}_1$, $\mathbf{t}_2$, and $\mathbf{t}_3$ fall on a circle of radius D centered at some point given by $\mathbf{t}_0$. See figure 3.4.

Lemma 2 *The following system as a function of ϕ has at most six distinct solutions, unless all ϕ are solutions.*

$$\begin{aligned}(\mathbf{t}_1(\phi) - \mathbf{t}_0)(\mathbf{t}_1^*(\phi) - \mathbf{t}_0^*) &= D^2 \\ (\mathbf{t}_2(\phi) - \mathbf{t}_0)(\mathbf{t}_2^*(\phi) - \mathbf{t}_0^*) &= D^2 \\ (\mathbf{t}_3(\phi) - \mathbf{t}_0)(\mathbf{t}_3^*(\phi) - \mathbf{t}_0^*) &= D^2\end{aligned}$$

Corollary 1 *For a given triple of feature matches, there are at most 6 values of ϕ where the three match circles intersect at the same point, unless the three match circles intersect at one point for all ϕ.*

Proof:

A circle of radius D containing the three points $\mathbf{t}_1$, $\mathbf{t}_2$, and $\mathbf{t}_3$ can be found by solving the following system of non-linear equations for ϕ and $\mathbf{t}_0 = u + iv$:

$$\begin{aligned}(\mathbf{t}_1(\phi) - \mathbf{t}_0)(\mathbf{t}_1^*(\phi) - \mathbf{t}_0^*) &= D^2 \\ (\mathbf{t}_2(\phi) - \mathbf{t}_0)(\mathbf{t}_2^*(\phi) - \mathbf{t}_0^*) &= D^2 \\ (\mathbf{t}_3(\phi) - \mathbf{t}_0)(\mathbf{t}_3^*(\phi) - \mathbf{t}_0^*) &= D^2\end{aligned}$$

By expanding then subtracting any two of the above equations we get

$$\mathbf{t}_i\mathbf{t}_i^* - \mathbf{t}_j\mathbf{t}_j^* - [\mathbf{t}_0(\mathbf{t}_i^* - \mathbf{t}_j^*) + \mathbf{t}_0^*(\mathbf{t}_i - \mathbf{t}_j)] = 0$$

From this we construct the linear system:

$$\begin{bmatrix} (\mathbf{t}_2^* - \mathbf{t}_1^*) & (\mathbf{t}_2 - \mathbf{t}_1) \\ (\mathbf{t}_3^* - \mathbf{t}_2^*) & (\mathbf{t}_3 - \mathbf{t}_2) \end{bmatrix} \begin{bmatrix} \mathbf{t}_0 \\ \mathbf{t}_0^* \end{bmatrix} = \begin{bmatrix} \mathbf{t}_2\mathbf{t}_2^* - \mathbf{t}_1\mathbf{t}_1^* \\ \mathbf{t}_3\mathbf{t}_3^* - \mathbf{t}_2\mathbf{t}_2^* \end{bmatrix}$$

The determinant of the above matrix is $(\mathbf{t}_2^* - \mathbf{t}_1^*)(\mathbf{t}_3 - \mathbf{t}_2) - (\mathbf{t}_2 - \mathbf{t}_1)(\mathbf{t}_3^* - \mathbf{t}_2^*)$ which is non-zero when $\mathbf{t}_1$, $\mathbf{t}_2$, and $\mathbf{t}_3$ are not colinear, thus the solution of this system is the unique circle of some radius on which $\mathbf{t}_1$, $\mathbf{t}_2$, and $\mathbf{t}_3$ fall. The case were the determinant is zero results when the three points lie on a circle of infinite radius. We can solve this for $\mathbf{t}_0$ using Cramer's rule:

$$\mathbf{t}_0 = \frac{\begin{vmatrix} (\mathbf{t}_2\mathbf{t}_2^* - \mathbf{t}_1\mathbf{t}_1^*) & (\mathbf{t}_2 - \mathbf{t}_1) \\ (\mathbf{t}_3\mathbf{t}_3^* - \mathbf{t}_2\mathbf{t}_2^*) & (\mathbf{t}_3 - \mathbf{t}_2) \end{vmatrix}}{\begin{vmatrix} (\mathbf{t}_2^* - \mathbf{t}_1^*) & (\mathbf{t}_2 - \mathbf{t}_1) \\ (\mathbf{t}_3^* - \mathbf{t}_2^*) & (\mathbf{t}_3 - \mathbf{t}_2) \end{vmatrix}}$$

which yields

$$\mathbf{t}_0 = \frac{(\mathbf{t}_2\mathbf{t}_2^* - \mathbf{t}_1\mathbf{t}_1^*)(\mathbf{t}_3 - \mathbf{t}_2) - (\mathbf{t}_3\mathbf{t}_3^* - \mathbf{t}_2\mathbf{t}_2^*)(\mathbf{t}_2 - \mathbf{t}_1)}{(\mathbf{t}_2^* - \mathbf{t}_1*)(\mathbf{t}_3 - \mathbf{t}_2) - (\mathbf{t}_3^* - \mathbf{t}_2^*)(\mathbf{t}_2 - \mathbf{t}_1)}$$

We seek ϕ and $\mathbf{t}_0$ such that this circle has radius D. So, substitute $\mathbf{t}_0$ into

$$(\mathbf{t}_1(\phi) - \mathbf{t}_0)(\mathbf{t}_1^*(\phi) - \mathbf{t}_0^*) = D^2$$

Now

$$(\mathbf{t}_1 - \mathbf{t}_0) = \frac{(\mathbf{t}_2 - \mathbf{t}_1)(\mathbf{t}_3 - \mathbf{t}_1)(\mathbf{t}_3^* - \mathbf{t}_2^*)}{(\mathbf{t}_2^* - \mathbf{t}_1^*)(\mathbf{t}_3 - \mathbf{t}_2) - (\mathbf{t}_2 - \mathbf{t}_1)(\mathbf{t}_3^* - \mathbf{t}_2^*)}$$

which leads to the equation

$$(\mathbf{t}_2 - \mathbf{t}_1)(\mathbf{t}_3 - \mathbf{t}_1)(\mathbf{t}_3^* - \mathbf{t}_2^*)(\mathbf{t}_2^* - \mathbf{t}_1^*)(\mathbf{t}_3^* - \mathbf{t}_1^*)(\mathbf{t}_3 - \mathbf{t}_2) +$$
$$D^2[(\mathbf{t}_2^* - \mathbf{t}_1^*)(\mathbf{t}_3 - \mathbf{t}_2) - (\mathbf{t}_2 - \mathbf{t}_1)(\mathbf{t}_3^* - \mathbf{t}_2^*)]^2 = 0$$

Recalling that $\mathbf{t} = \mathbf{p}_d - e^{i\phi}\mathbf{p}_m$, we seek those ϕ for which the above equation is true. Multiplying the above equation by $e^{3i\phi}$ will not change its roots, and inspection shows the result is a 6^{th} degree polynomial in $e^{i\phi}$. There are at most six distinct solutions to this, unless any ϕ is a solution when all the coefficients are zero. Note that since the above equation is purely real, the coefficients to powers of $e^{i\phi}$ and $e^{-i\phi}$, respectively, are complex conjugates; thus, the equation is a 3^{rd} degree trigonometric polynomial in $\cos(\phi)$ and $\sin(\phi)$. □

The previous several lemmas provide the basis to prove the following central lemma. The intuition is that as we watch a match area evolve with ϕ, at some point in its life it will be bordered by a 2-way or 3-way match-circle intersection. Because the match area traces out the surface of the intersection volume, these points fall on the bounding surface of the intersection volume. First, note a simple property of locally maximal intersection areas that concerns their evolution.

Claim 1 *An intersection area of locally maximal value is a convex region of the* (u, v) *plane.*

Proof: The boundary of any intersection area is composed piecewise of circular arcs, joined at the places where two circles intersect. Suppose we have a maximal intersection area that is not convex, that is, one of the circular segments curves into rather than out of the region. Then at some point along this curve segment we could step across into a region contained within the same match circles, but to which we have added $f(< m_i, d_j >, T)$ corresponding to the circle just entered and have thus increased the value of $F(T)$, a contradiction. □

Lemma 3 *A maximal intersection volume is bordered by a point where the surfaces of either two or three match regions intersect, unless it is composed of a single match region.*

Proof:

Recall that for a particular intersection volume, its extent in ϕ is given by $[\phi_a, \phi_b] = \bigcap_{i,j}[\phi_{a_{i,j}}, \phi_{b_{i,j}}]$ for match regions associated with matches $< m_i, d_j >$ forming the intersection volume.

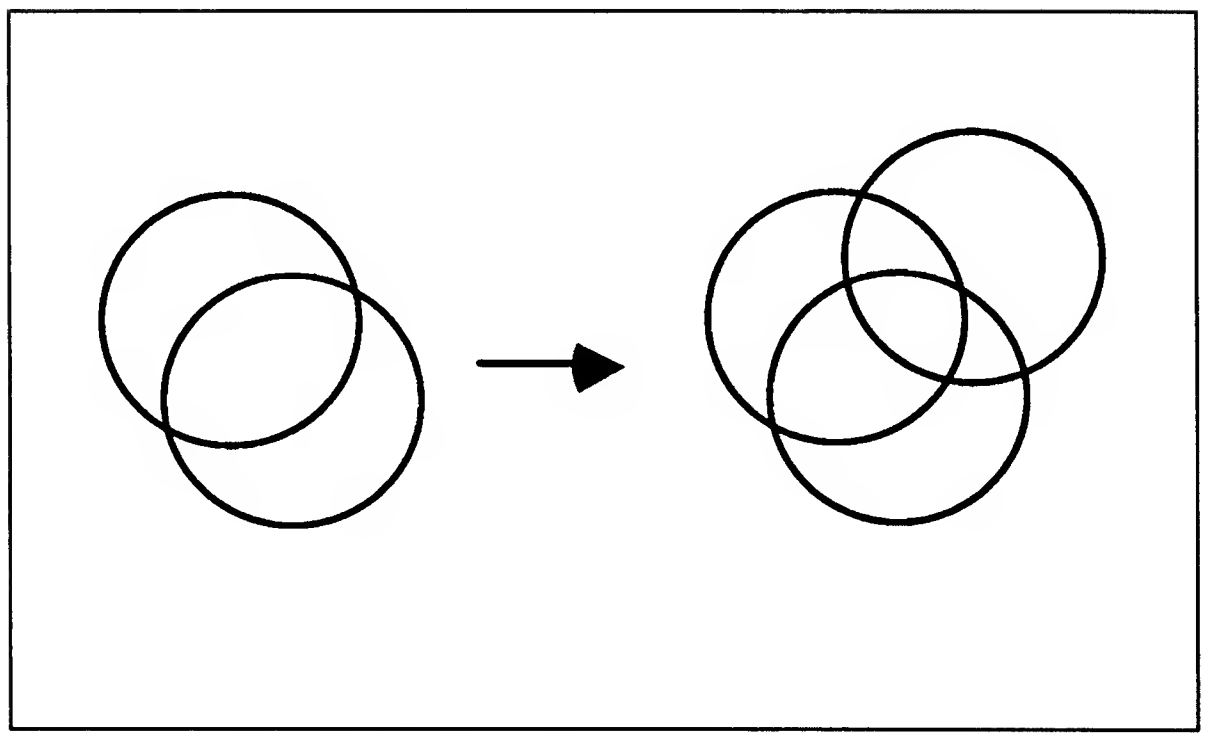

Figure 3.5: At the end of the intersection volume where $\phi = \phi_b$ or $\phi = \phi_e$, we must look at places where pairs of match circles intersect. This is one of the places where a match circle appears or disappears.

In the special case the intersection volume is composed of a single match region existing over a range $[\phi_a, \phi_b]$, then the center points of the match circle at ϕ_a or ϕ_b both border the volume. There are no more than $2(mn)$ such points.

Next, consider the first case mentioned above where over the range of $\phi \in [\phi_a, \phi_b]$, the area of the intersection area is non-zero, but is composed of the intersection of two or more match regions. In this case, the intersection points of the match circles at ϕ_a and ϕ_b border the volume. Since there are mn match circles, $\binom{mn}{2}$ pairs of match circles, and at most 4 such points per pair of match regions, there are less than $2(mn)^2$ such points. See figure 3.5.

In the second case when the intersection area does not exist at ϕ_a or ϕ_b, but exists for some $\phi \in (\phi_a, \phi_b)$, the above method fails. In this case, because of the fact described in claim 1, the area evolves from a single point which is the intersection of two match circles, or three or more match circles. If the area evolves from the intersection of two match circles there exists a ϕ where the centers of the two match circles are exactly a distance $2D$ apart. From the lemma 1 there are no more than two ϕ where this is true. Because there are $\binom{mn}{2}$ pairs of match circles, there are less than $(mn)^2$ such points. See figure 3.6. If the area evolves from the intersection of three match circles there is a point where 3 match circles

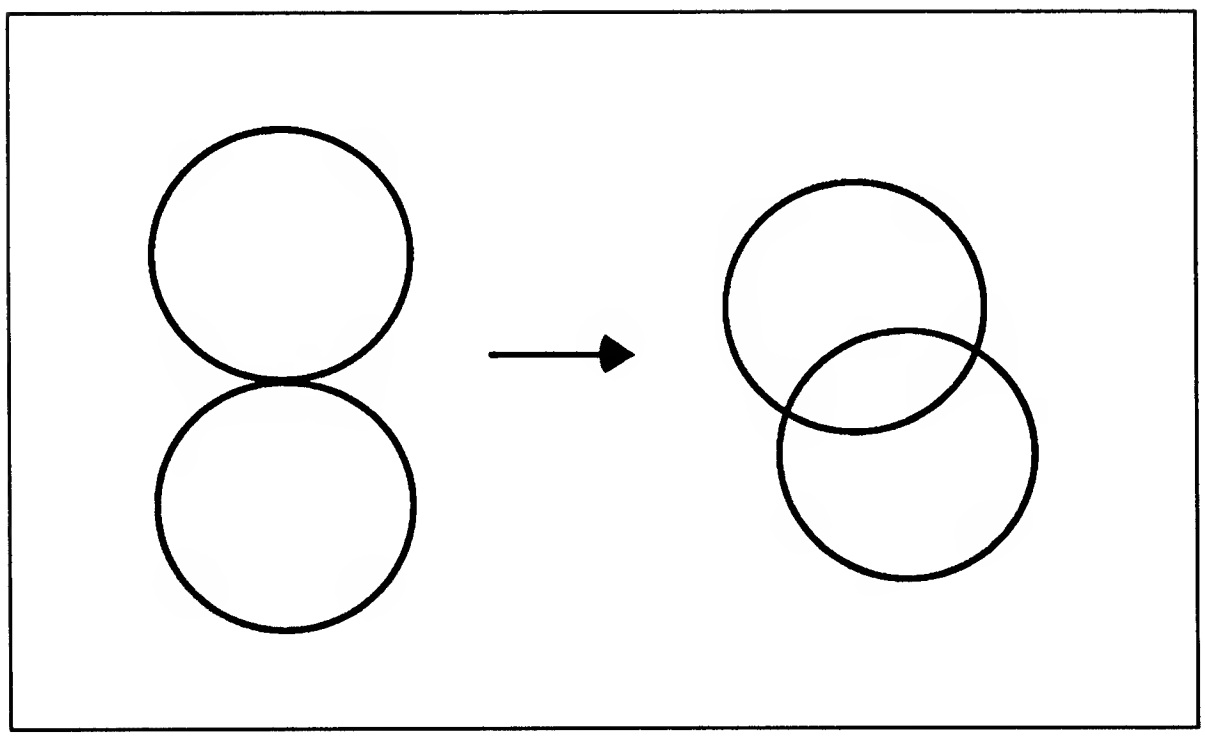

Figure 3.6: Evolution of a new intersection region from the point of intersection of two match circles.

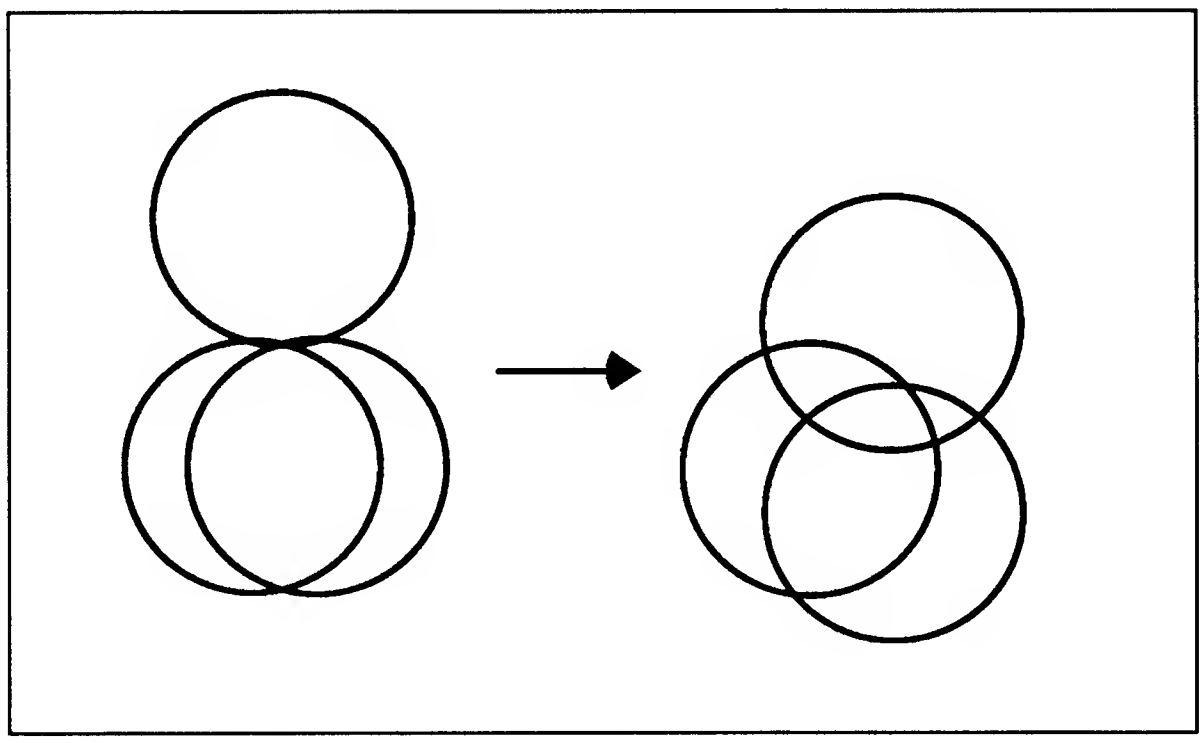

Figure 3.7: Evolution of a new intersection region from the point of intersection of three match circles.

intersect. From corollary 1 there are no more than six ϕ where this is true. Since there are $\binom{mn}{3}$ triples of match circles, there are less than $(mn)^3$ such points. See figure 3.7.

Note that the case where the circles intersect for all ϕ is of no importance since in this case no intersection area evolves from that particular point. □

The last lemma handles the case where the the match-region is locally maximal in value. The case where match-regions are not locally maximal is exactly the same with the addition of one special case for the birth and death of a match-area, where two match circles are superimposed and then move off of one another, or the reverse. In this case there are two special points that must be considered. Let $\mathbf{t}_1$ and $\mathbf{t}_2$ describe the centers of these circles. Derive the quantity $\frac{\partial}{\partial\phi}(\mathbf{t}_1 - \mathbf{t}_2)$ at the ϕ where $\mathbf{t}_1 = \mathbf{t}_2$. The two points of intersection of the superimposed match circles with the line through their common center in the direction perpendicular to this vector are critical points at the birth or death of a match region. Because there are $\binom{mn}{2}$ paris of match circles, and two such points for each, there are less than m^2n^2 such points.

3.3.4 An Algorithm for Matching Hypothesis

From the above theorem we see that there are $O(m^3n^3)$ points that need to be examined to ensure finding one that borders each maximal intersection volume. Following is an outline of a brute force algorithm for hypothesis generation.

- Form all feature matches.
- Find all points where where two match regions intersect, and where three match regions intersect. There are $O(m^3n^3)$ such points.
- For each such point, query each match region to find those that contain each point.
- Select those points which fall at an intersection region with a maximal value of $F(T)$

This procedure requires the numerical solution of a 3^{rd} degree trigonometric polynomial in $\cos(\phi)$ and $\sin(\phi)$, or alternatively the solution to a regular 6^{th} degree polynomial with complex coefficients. A more detailed algorithmic development of this idea is provided in chapter 4.

3.4 Uniform Sampling

Overview

Hypothesis generation has been cast as searching **TPS** for optimal intersection volumes. The previous section introduced a method of finding optimal intersection volumes based on computing the location of points lying in the bounding surface of the intersection volumes, and provided insight into the nature of the search space. This section introduces another approach to finding optimal intersection volumes, based on sampling **TPS** to find a point that lies in an optimal intersection volume.

The basic intuition is that given any point T in **TPS**, it is simple to determine the value of $F(T)$ at T by computing $f(< m, d >, T)$ for each match. A systematic approach would be to sample the entire parameter space on a regular grid, to find some such sample points which fall inside optimal intersection volumes. This is a finite but extremely large set of sample points. We can reduce the number of sample points considerably, however, by noting that for a particular feature match we need only consider those sample points which actually fall inside its match region. Let K represent the number of uniformly spaced sample points in **TPS** which fall inside a match-region. In this case we need only consider Kmn transformation sample points.

An advantage of sampling on a uniform grid in **TPS** is that the containment of each sample point in all the match regions need not be explicitly computed. Instead, for each match region the set of sample points falling in it is computed, and these sample points then simply sorted into groups according to the sample point. For each group, $F(T)$ is computed. If the sampling intervals are fine enough, at least one of the sample points will fall in each of the optimal intersection volumes, achieving the goal. So the success of this approach depends on a sample point falling inside the optimal intersection volume, otherwise we will never find this region. The sampling intervals in rotation and translation must be small enough to ensure this. As we shall see, there is a fundamental tradeoff between accuracy and complexity: the finer the sampling interval, the greater the number of transformations to be evaluated.

3.4.1 An Algorithm for Matching Hypothesis

The basic outline of an algorithm for uniform transformation sampling consists of first forming all feature matches, and computing the transformation sample points on a the uniform grid which fall inside each match region. The transformation sample points thus

constructed are then sorted, and for each particular sample point T, the piecewise constant metric $F(T)$ is computed for each group of equal transformation sample points, and the transformation sample points yielding maximal values of $F(T)$ are taken as initial hypotheses. These hypotheses are then refined to determine the best estimates of the transformations T for the matchings initially hypothesized, and the refined hypotheses then verified. The following sections provide a more formal basis for uniform sampling.

3.4.2 A Formal Analysis of Uniform Sampling

The key issue with uniform sampling is to determine lower bounds on the sampling intervals required to ensure sampling within the optimal intersection volume. This section develops a useful probabilistic model of the structure of **TPS** under the bounded uncertainty constraint, which will provide a formal basis for the uniform transformation sampling approach.

The basic intuition is as follows. By definition, each match region in **TPS** due to correctly matched features contains the *correct* rotation and translation. Thus, the optimal intersection volume is a region contained in all the correct match regions, which also contains the correct transformation. In the following analysis, we will fit a cylinder of radius r and length 2ξ, with axis parallel to the ϕ axis of **TPS** and centered at the correct transformation, such that it is completely contained in the optimal intersection volume. This cylinder provides a lower bound on the intersection volume. By choosing the sampling intervals such that at least one will fall inside such a cylinder, we can be sure to find the optimal intersection volume. The more the measured pose of a non-spurious image feature deviates from its correct pose, the more the center of its associated correct match-region deviates in **TPS** from the correct transformation, making the region of optimal intersection smaller. The idea of the following analysis is to formulate a probabilistic lower bound on the size of the cylinder just described, based on the probabilistic deviations of the measured pose of non-spurious image features from their correct pose.

The Probabilistic Structure of Parameter Space

As before, I will consider point features. In the following analysis, assume the existence of $\mathbf{t}_0$ and ϕ_0 representing the *correct* translation and rotation, respectively. By definition of pose uncertainty, all correctly matched features define a match-region which contains the *correct* rotation ϕ_0 and translation $\mathbf{t}_0$. As mentioned above, the size of the optimal intersection volume depends on the extent to which correct match regions overlap each

other, that is, the extent to which the positions in **TPS** of correct match-regions deviate from the correct transformation. If there were no image feature pose error, the correct transformation would be in the center of the each correct match region match region. The deviation of a correct match-region from the correct transformation is directly related to the deviation of the measured pose of the image feature from the correct pose.

Deviation from the Correct Transformation

The probability distribution for a random variable x will be denoted $g_x(x)$, and $P_{x \leq x_0}$ denotes the probability x is less than or equal to some constant x_0. Assume that the deviation of the measured pose of an image feature from the actual pose is characterized by the distribution $g_{\theta_e}(\theta_e)$ where $\theta_e = \theta_{d_0} - \theta_d$ is the difference between the correct orientation of an image feature and the measured value, respectively. Similarly, assume $g_\rho(\rho)$ characterizes the distribution of $\rho = |\mathbf{t}_0 - \mathbf{t}|$, the absolute distance in $u - v$ space between the correct translation and the translation calculated from the measured feature, and taken in the plane $\phi = \phi_0$ where ϕ_0 is the *correct* rotation. The distributions $g_{\theta_e}(\theta_e)$ and $g_\rho(\rho)$ could be calculated based on models of the mechanisms producing sensing errors, such as noise and the effects of feature extraction. An alternative is to calculate them empirically from actual sensor measurement.

As mentioned above, each correct match-region contains the correct transformation ϕ_0, $\mathbf{t}_0$. The idea of this analysis is to determine the size of a cylinder centered at ϕ_0, $\mathbf{t}_0$, and completely contained inside the optimal intersection volume, thus bounding it from below. To approach this, first such a cylinder centered at ϕ_0, $\mathbf{t}_0$ is fit inside each correct match-region. Call this cylinder a *match-cylinder*. The size of these match-cylinders is characterized probabilistically, and then these individual match-cylinders are related together to form a lower bound on the optimal intersection volume.

The Radius of a Match Cylinder

Again, note that a match cylinder is centered at ϕ_0, $\mathbf{t}_0$, and its axis of rotation aligned with the ϕ axis of **TPS**. First we will consider lower bounds on the extent of a match cylinder in the translational dimensions of **TPS**. Consider the match region defined by a particular correct feature match, and the match circle formed by a slice through **TPS** at $\phi = \phi_0$. Because this match region is due to correctly matched features, it contains the *correct* translation $\mathbf{t}_0$. Note that a circle of radius $r = D - \rho$ centered at $\mathbf{t}_0$ is completely contained within the match circle. See figure 3.8. Call this circle C_r. We can easily derive

the probability distribution of the radius r of C_r from $g_\rho(\rho)$: $g_r(r) = g_\rho(D-r)$, because in general if $x = a - y$ then

$$g_y(y_0) = \frac{d}{dy_0} P_{y \le y_0} = \frac{d}{dy_0} P_{a-y_0 \le x} = \frac{d}{dy_0} \left(1 - \int_{-\infty}^{a-y_0} g_x(x)dx\right) = g_x(a - y_0)$$

The probability that, for a given correct match region, the circle C_r taken at a cross section $\phi = \phi_0$ has radius $r \le r_0$ is given by:

$$P_{r \le r_0} = \int_{-\infty}^{r_0} g_r(r)dr.$$

We are aiming toward determining the size of a match-cylinder. Note that the match cylinder cannot have the circle C_r as cross section because this is only true at $\phi = \phi_0$. As ϕ moves away from ϕ_0, the minimum distance from $\mathbf{t}_0$ to the edge of the match circle can decrease. Recall from section 3.3 that as ϕ varies, the match circle moves according to $\mathbf{t} = \mathbf{p}_d - e^{i\phi}\mathbf{p}_m$ for a particular feature match $< m, d >$. We can view the position of the match circle in the $u - v$ plane as a function of ϕ. The velocity, $\frac{\partial \mathbf{t}}{\partial \phi}$, at which a match circle moves in the $u - v$ plane as ϕ varies is proportional to $|\mathbf{p}_m|$, the distance of its center from the center of rotation. We can bound this velocity as follows: If the maximum dimension of the image is I, then we can shift the model features such that $|\mathbf{p}_m| \le \frac{\sqrt{2}I}{2}$ for all model features. In this case the maximum velocity with respect to ϕ of a match circle is $|\frac{\partial \mathbf{t}}{\partial \phi}| \le \omega = \frac{\sqrt{2}I}{2}$. Thus, the circle C_r with radius r_0 at $\phi = \phi_0$ may shrink as ϕ varies from ϕ_0, but its radius can change at a rate no faster than ω. A lower bound for the radius of a new circle $C_{r'}$ centered at $\mathbf{t}_0$ as ϕ varies from ϕ_0 is then given by $r' = r_0 - \omega|\phi_0 - \phi|$, for $|\phi_0 - \phi| \le \frac{r_0}{\omega}$. So at any position ϕ, a circle of radius r' is within the match region in the $u - v$ dimensions. The value r' is the radius of the match-cylinder.

The Length of a Match Cylinder

Next I will characterize the extent of a match cylinder in the ϕ direction of **TPS**. The probability distribution $g_{\theta_e}(\theta_e)$ characterizes the distribution of the error in orientation measurement for an image feature, where $\theta_e = \theta_{d_0} - \theta_d$. Because the measured relative rotation ϕ is given by $\phi_{dm} = \theta_d - \theta_m$, and by definition $\phi_0 = \theta_{d_0} - \theta_m$, we have that

$$\phi_{dm} = (\theta_{d_0} - \theta_m) - \theta_e = \phi_0 - \theta_e.$$

Defining $\phi_e = \phi_0 - \phi_{dm}$ to be the error in relative rotation, we see $\phi_e = \theta_e$, thus $g_{\phi_e}(\phi) = g_{\theta_e}(\phi)$.

The total extent of a match region in the ϕ dimension is 2Θ. Let $\xi = \Theta - |\phi_0 - \phi_{dm}| = \Theta - |\phi_e|$ be the distance from ϕ_0 to the closest edge of a correct match region in the ϕ direction. Let 2ξ be the length of the match-cylinder, thus the match-cylinder is symmetric about ϕ_0, t_0, and extends to the boundary of the match-region. We now calculate the distribution $g_\xi(\xi)$ characterizing the length of the match-cylinder.

With $\xi = \Theta - |\phi_e|$, $P_{\xi \leq \xi_0}$ is the sum of two probabilities: $P_{\Theta + \phi_e \leq 0}$ for $\phi_e < \phi$ and $P_{\Theta - \phi_e \leq 0}$ for $\phi_e \geq \phi$. Thus,

$$P_{\xi \leq \xi_0} = \int_{-\infty}^{-(\Theta - \xi_0)} g_{\phi_e}(\phi) d\phi + \int_{(\Theta - \xi_0)}^{\infty} g_{\phi_e}(\phi) d\phi$$

and

$$g_\xi(\xi_0) = \frac{d}{d\xi_0} P_{\xi \leq \xi_0} = g_{\phi_e}(\xi_0 - \Theta) + g_{\phi_e}(\Theta - \xi_0)$$

If $g_{\phi_e}(\phi)$ is symmetric, $g_\xi(\xi) = 2 g_{\phi_e}(\Theta - \xi_0)$ The probability that, for a given correct match region, the length of the match-cylinder is $2\xi \leq 2\xi_0$ is given by:

$$P_{\xi \leq \xi_0} = \int_{-\infty}^{\xi_0} g_\xi(\xi) d\xi.$$

The Optimal Intersection Volume

The above results can be combined to provide a probabilistic lower bound on the size of the optimal intersection volume. For a given match-region, the probability that a match cylinder has radius $r' \leq r_0 - \omega\xi_0$ and length at least $2\xi_0$ is given by

$$P_{\xi > \xi_0, r > r_0} = \int_{\xi_0}^{\infty} \int_{r_0}^{\infty} g_\xi(\xi) g_r(r) dr d\xi = \left(1 - \int_{-\infty}^{\xi_0} g_\xi(\xi) d\xi\right) \left(1 - \int_{-\infty}^{r_0} g_r(r) dr\right) = P_{\xi > \xi_0} P_{r > r_0}$$

where $P_{x > x_0} = 1 - P_{x \leq x_0}$, and it is assumed that the uncertainty in orientation is independent of the uncertainty in position. Assume there are N correctly matched image and model features possible, that is, there are N of the model features visible in the image. Let $P_{\xi_0 r_0} = P_{\xi > \xi_0} P_{r > r_0}$. Assuming the measurement errors are independent, the probability that k of N match cylinders have length $2\xi > 2\xi_0$ and radius $r > r_0$ is given by by the binomial distribution:

$$B_{P_{\xi_0 r_0}}(k) = \binom{N}{k} P_{\xi_0 r_0}^k (1 - P_{\xi_0 r_0})^{N-k} \quad 0 \leq k \leq N$$

Let P be the probability of the event that for a single match-region a circular cylinder of radius r' and length $2\xi_0$ is completely contained in the match-region. $P > P_{\xi_0 r_0}$ for a

match-region since the cylinder is not necessarily symmetric about ϕ_0 and can be longer than $2\xi_0$.

Construct the cumulative probability $C_P(k)$ that for k or less match-regions a circular cylinder of radius r' and length $2\xi_0$ is completely contained in the match regions where

$$C_P(k) = \sum_{i=0}^{k} B_P(i) = \sum_{i=0}^{k} \binom{N}{i} P^i (1-P)^{N-i}$$

and the probability that this is true for k or more match regions is $1 - C_P(k-1)$. Because $P > P_{\xi_0 r_0}$, $1 - C_P(k-1) > 1 - C_{P_{\xi_0 r_0}}(k-1)$, providing a lower bound on the probability the described cylinder is contained in k or more correct match-regions.

We can use the following result to characterize the behavior of the probability as N grows: It can be shown[19] that $C_P(\lfloor \alpha N \rfloor) \leq e^{-2N(\alpha - P)^2}$, $\alpha \leq P$ which is the case considered in the example to follow. This means that as N increases, the probability that $(1 + \lfloor \alpha N \rfloor)$ or more match-regions contain a cylinder of the given size approaches 1.

In summary, this analysis allows us to obtain lower bounds on the probability the optimal intersection volume will be at least as big as a cylinder of radius r' and length $2\xi_0$. As an approximation to finding the optimal intersection volume, the analysis also provides a bound on the probability of such a cylinder contained in k of the N correct match regions.

Uniform Sampling Intervals

The previous analysis derived a lower bound on the probability that a cylinder of radius r' and length $2\xi_0$ is completely contained in k out of N match regions. The sampling intervals are chosen so that at least one point will fall in this cylinder. In order to be sure we sample in the translation dimensions within a circle of radius $r' = r_0 - \omega\xi_0$ we must have $\delta t \leq \sqrt{2}r'$. In the ϕ dimensions, we must have that $\delta\phi \leq 2\xi_0$. A general strategy for determining sampling intervals is to first pick an acceptable probability bound on finding a correct matching of size k or larger. This determines a locus of points (ξ_0, r_0). The appropriate sampling intervals are given by these two values.

Discussion

The preceding analysis makes several assumptions. Most importantly, it is assumed that the uncertainty in image feature pose can be characterized by a probability distribution. Secondly, it is assumed that the deviations in orientation and position from the correct

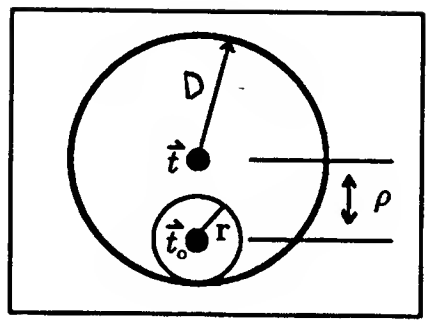

Figure 3.8: A circle of radius r centered at $\mathbf{t}_0$ fits inside the match circle.

values are independent of one another, characterized by $g_\rho(\rho)$ and $g_{\theta_e}(\theta_e)$. Finally, it is assumed that the deviation for different features is independent.

Assuming that pose uncertainty can be modeled probabilistically is reasonable. There are many mechanisms which cause uncertainty, and characterizing them deterministically is probably fairly difficult. The probabilistic model allows the formulation of an analysis of the system which, while not entirely accurate, lends some justification to the approach. Even if we allow the characterization of pose uncertainty probabilistically, the problem of justifying particular probability distributions remains. It is important to note, however, that the preceding arguments all depend on the integrals of the distributions $g_\rho(\rho)$ and $g_{\theta_e}(\theta_e)$, and not on the distributions themselves. Thus the effects of the choice of the particular distributions is lessened.

The assumption that the orientation uncertainty and the position uncertainty are independent is probably incorrect. If a feature is derived from the boundary contour of an object, any process which deforms the contour by changing the position of a particular curve segment will very likely also change the orientation at that point. Finally, the assumption that the individual errors are independent is reasonable for features distant from one another, but probably unreasonable for nearby features. Any mechanism causing a feature's measured position to vary could very likely affect nearby features in the same way, invalidating the independence assumption for some features.

It is important to note that this analysis provides a simple, extremely loose lower bound on the sampling intervals δt and $\delta\phi$ for three main reasons. First, the region of overlap of match circles does not have circular cross section, but is actually larger than the cross section of the cylinder fit inside it. Second, the extent of the overlap of match regions in the ϕ dimension is asymmetric about the correct rotation ϕ_0, including more than the region of length 2ξ considered in the analysis. Third, the rate that match circles move,

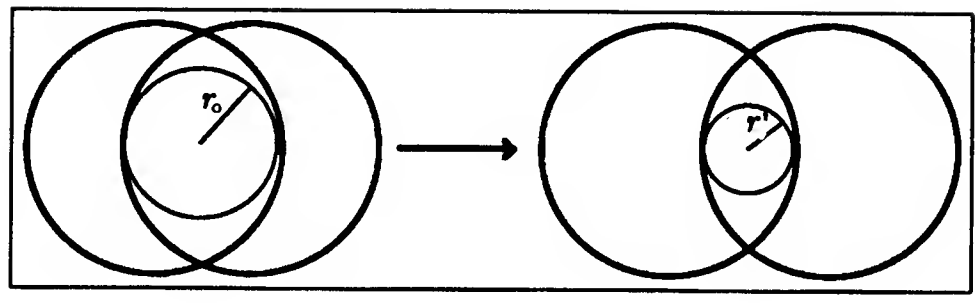

Figure 3.9: As ϕ moves away from the correct value ϕ_0, the circle C_r of radius r_0 centered at $\mathbf{t}_0$ shrinks to radius r'.

and that the fitted cylinder shrinks in radius, is usually much less than the upper bound of $\omega = \frac{\sqrt{2}I}{2}$ used in the analysis. For these three reasons, the optimal intersection volume is in reality much larger than the lower bound provided by the analysis, and thus the bounds on sampling intervals and on probabilities are very conservative, and in practice we would expect that near optimal match regions would be found with high probability using coarser sampling intervals.

The assumptions made are very likely not completely correct, and this analysis is not intended to be a rigorous justification of the method. Instead, it is intended to make plausible the idea of uniform transformation sampling.

An Example

Here is an example to give a more concrete sense of the implications of the previous analysis. Included are two examples with two different distributions for each of $g_{\theta_e}(\theta_e)$ and $g_\rho(\rho)$. In the first case $g_{\theta_e}(\theta_e)$ is modeled as Gaussian and $g_\rho(\rho)$ as a Gaussian defined for $\rho \geq 0$. In the second case, all position deviations and all orientation deviations are taken to be equally likely.

Assume that for $0 \leq \rho \leq D$, $g_\rho(\rho)$ is proportional to a Gaussian distribution:

$$g_\rho(\rho) \propto e^{-\frac{\rho^2}{2\sigma_\rho^2}}$$

and 0 elsewhere. Similarly assume $g_{\theta_e}(\theta_e)$ is proportional to a Gaussian for $-\Theta \leq \theta_e \leq \Theta$ and 0 elsewhere.

Let the standard deviation of the Gaussians for $g_{\theta_e}(\theta_e)$ and $g_\rho(\rho)$ be $\frac{\Theta}{2}$ and $\frac{D}{2}$ respectively. The following table shows an example of the parameters and probabilities for this case.

Θ	D	$\delta\phi$	δt	$P_{r_0\xi_0}$	ω	N	k	$C_k(k)$
12 deg	5 pix	.3323 deg	.7425 pix	.8660	$\frac{256}{\sqrt{2}}$	56	42	.0140

Thus we see that with probability 99%, 43 or more of 56 correct match regions will be found when using sampling intervals of .3323 degrees and .7425 pixels, for a 256^2 image.

As a second example, assume that any error within the uncertainty bounds is equally likely. In this case we have that

$$P_{\rho\leq\rho_0} = \int_0^{\rho_0} g_\rho(\rho) = \frac{\pi\rho_0^2}{\pi D^2}$$

so

$$g_\rho(\rho) = \frac{2\rho}{D^2}$$

thus

$$P_{r_0} = \int_0^{r_0} g_\rho(D-r)dr = \frac{r_0(2D-r_0)}{D^2}$$

also

$$P_{\xi\leq\xi_0} = \int_0^{\xi_0} g_\xi(\xi) = \frac{\xi_0}{\Theta}$$

considering a specific example:

Θ	D	$\delta\phi$	δt	$P_{r_0\xi_0}$	ω	N	k	$C_k(k)$
12 deg	5 pix	.0717 deg	.1603 pix	.8660	$\frac{256}{\sqrt{2}}$	56	42	.0140

Thus we see that with probability 99%, 43 or more of 56 correct match regions will be found when using sampling intervals of .0717 degrees and .1603 pixels, for a 256^2 image. The Gaussian distributions are probably more realistic because we might expect the deviations of position and orientation to be distributed near zero, rather than being uniformly distributed.

The equation introduced above, $C_P(\lfloor\alpha N\rfloor) \leq e^{-2N(\alpha-P)^2}$, $\alpha \leq P$ provides a simple bound on the probability that more than $\lfloor\alpha N\rfloor$ match-cylinders are above a given size. For example, defining β to be the probability that more than $\lfloor\alpha N\rfloor$ of the correct match regions contain the a match-cylinder of the given size, and given N and P, α is given by:

$$\alpha \geq P - \sqrt{\frac{\lg(1-\beta)^{-1}}{2N}}.$$

Taking a specific example, say $N = 56$ and $P = .8660$ as above, and we want $\beta \geq .99$, then $\alpha \geq .66$, and thus with probability 99%, over 66% of the correct match regions will completely contain the given cylinder. These bounds improve as N or P grows, providing a loose lower bound.

Extended Features

The previous analysis using point features provides a lower bound for the case of extended features such as line segments. For a particular feature match at a particular rotation there is a region of possible translations. The length of this region is a function of the position uncertainty and the difference in the lengths of the model and image features. An equivalent situation is given when we subtract the length of the image feature from both the model and the image features. Assuming the image feature is less than or equal in length to the model feature, we then have an image feature which is a point paired with a model feature which is either a point or a line. In the point-point feature case, the intersections of match regions had cross sections that were intersections of circles. In this case the cross section of intersections will be intersections of circles or circles and rectangles with hemispherical ends. It is easy to see that in this case the amount of overlap of match regions is equal to or greater than the case where only point features are involved. Thus, the point feature analysis provides a very loose lower bound on the performance of the technique when using extended features.

Handling Unknown Scale

In the formal analysis above we saw that as the rotation component, ϕ, of the transformation varies from the correct rotation ϕ_0, the region in $u - v$ space where the correct match circles overlap gets smaller. If the amount ϕ varies is small enough we can determine a lower bound on the size of this region. Appropriate choice of the rotation sampling interval limits the amount ϕ varies, and appropriate choice of the translational sampling interval then ensures that some sample point will fall in the region of optimal intersection. The situation is analogous when scale is unknown, and the range of possible scales is sampled.

When scaling is part of the transformation, the transformation between an image feature and a model feature consist of a rotation, followed by a scaling and a translation. The translation required to align two features is given by $\mathbf{t} = \mathbf{p}_d - se^{i\phi}\mathbf{p}_m$ were s is the scale factor. The overlap of the match regions in the ϕ dimension is unchanged by variations in

s, but the correct match circles will move relative to one another at a rate given by

$$|\frac{\partial \mathbf{t}}{\partial s}| = |\mathbf{p}_m| \leq \omega$$

where ω again provides a bound on the velocity of the match circles as before. The previous analysis provided a cylindrical region of radius r' and length $2\xi_0$ as a lower bound on the optimal intersection volume. If the scale parameter, s, is allowed to vary from the correct value s_0, the radius of this cylinder will shrink at the same rate as is the case when ϕ varies. So the radius of the new cylinder used as a lower bound to the optimal intersection volume as s varies from s_0 is given by $r'' = r' - \omega|s_0 - s|$. When the sampling interval in the scale dimension is δs, we have that $r'' \geq r' - \frac{\omega \delta s}{2}$. Once a scale sampling interval is chosen, we choose δt as before so that we are sure to sample in a cylinder of radius r''.

Adding the sampling in the scale dimension significantly increases the number of sample points we consider per match region. Unlike the case for translation and rotation samples, a single feature match does not constrain the possible scale samples. Without an a priori bound on the range of the scale s, we may have to consider a large number of scale samples. Recognition techniques which consider pairs of feature matches do not have this problem, because a pair of matches greatly constrains the range of possible scales to consider for those matches. Thus while asymptotically faster, scale sampling may be more complex in practice than an algorithm of $O(m^2n^2)$ that considers all pairs of feature matches in order to handle scale.

3.5 Probabilistic Sampling

A third technique for finding the optimal intersection volume is to sample **TPS** at randomly selected points. The idea behind probabilistic sampling is that if k random sample points are chosen within each of the N correct match regions, with some probability, at least one of the random points chosen will fall in the optimal intersection volume. This section develops a lower bound to this probability.

3.5.1 An Algorithm for Matching Hypothesis

Hypothesizing matchings basically consists of first forming all feature matches, and computing k random transformation sample points falling inside each match region. For each such sample point, the number of match regions which contain it is computed. From this the value of the piecewise constant definition of $F(T)$ can be computed at each sample

point. The straightforward approach is to form all pairs of sample points with match regions, compute the value of $f(< m, d >, T)$ for each sample point T and all matches $< m, d >$, then organize the results according to T, and compute the value of $F(T)$. The optimal values of $F(T)$ are taken, accomplishing hypothesis generation. These hypotheses are then refined to determine the best estimates of the transformations T for each matching initially hypothesized. These form the refined hypotheses which are then verified.

3.5.2 A Formal Analysis of Probabilistic Sampling

Recall from the analysis of the uniform sampling technique that for a correct match region, there is a circle of radius r_0 in the plane $\phi = \phi_0$, centered at $\mathbf{t}_0$ which is exactly contained within the match region. As ϕ varies from ϕ_0, this circle maintains radius at least $r' = r_0 - \omega|\phi - \phi_0|$. Thus as ϕ varies, this circle traces out a circular cone, providing a lower bound on the volume of an area centered at the correct transformation and contained within the match region. For convenience, call this region a *match-cone*. Depending on the quantity ξ, defined earlier as the distance between ϕ_0 and the edge of the match region in the ϕ dimension, the match-cone may not be a complete cone on one side.

The first step in the analysis is to derive the probability distribution for the quantity V, the volume of a match-cone. From this we can derive the probability distribution for the minimum of V over all N correct match regions. This then, is the distribution characterizing a lower bound on the volume of the optimal intersection volume.

The Volume of Match-Cones

For convenience we will make the gross approximation that the match-cone is symmetric about ϕ_0, $\mathbf{t}_0$. Let $\xi' = |\phi_0 - \phi|$, and $\xi = \Theta - |\phi_0 - \phi_{dm}| = \Theta - |\phi_e|$ as before. The cross section of the match-cone has radius $r' = r_0 - \omega\xi'$ as ϕ moves away from ϕ_0. So the volume of the match-cone is given by

$$V(r, \xi) = 2\int_{\xi'=0}^{\xi} \pi r'^2 d\xi' = 2\int_{\xi'=0}^{\xi} \pi(r_0 - \omega\xi')^2 d\xi' = \frac{2\pi}{3\omega}(r^3 - (r - \omega\xi)^3), \qquad \xi \leq \frac{r}{\omega}$$

and

$$V(r, \xi) = \frac{2\pi}{3\omega} r^3, \qquad \xi > \frac{r}{\omega}.$$

Now, $g_V(V_0) = \frac{d}{dV_0} P_{V \leq V_0}$, and to compute $P_{V \leq V_0}$ we must integrate the combined distribution $g(r, \xi)$ over the region $V \leq V_0$. Inspection of the two partial derivatives V_ξ and V_r

shows that $V(r,\xi)$ is monotonic in each of these variables, and some algebra shows that

$$\xi = \frac{r - (r^3 - \frac{3\omega V_0}{2\pi})^{\frac{1}{3}}}{\omega}, \qquad \xi \leq \frac{r}{\omega}$$

so,

$$P_{V \leq V_0} = \int_0^{(\frac{3\omega V_0}{2\pi})^{\frac{1}{3}}} g_r(r)dr + \int_{r=(\frac{3\omega V_0}{2\pi})^{\frac{1}{3}}}^{\infty} \int_{\xi=0}^{\frac{r-(r^3-\frac{3\omega V_0}{2\pi})^{\frac{1}{3}}}{\omega}} g_{r,\xi}(r,\xi)d\xi dr$$

and assuming that $g_{r,\xi}(r,\xi) = g_r(r)g_\xi(\xi)$,

$$g_V(V_0) = \frac{d}{dV_0}P_{V \leq V_0} = g_r(\left(\frac{3\omega V_0}{2\pi}\right)^{\frac{1}{3}})\frac{\omega}{2\pi}\left(\frac{3\omega V_0}{2\pi}\right)^{-\frac{2}{3}}\left(1 - \int_0^{(\frac{3\omega V_0}{2\pi})^{\frac{1}{3}}} g_\xi(\xi)d\xi\right).$$

The Volume of the Optimal Intersection Volume

Let V_{opt} denote the volume of the smallest of the match-cones, which is the cone contained in all correct match regions, and let V_1, V_2, ... V_N be the volumes of the individual match-cones for each of the N correct match regions. Then $V_{opt} = \min(V_1, V_2, \ldots V_N)$, and as before

$$g_{V_{opt}}(V_0) = \frac{d}{dV_0}P_{V_{opt} \leq V_0} = \frac{d}{dV_0}\left(1 - P_{V_{opt} > V_0}\right)$$

Now,

$$P_{V_{opt} > V_0} = \int_{V_1=V_0}^{\infty}\int_{V_2=V_0}^{\infty}\cdots\int_{V_N=V_0}^{\infty} g_V(V_1)\cdots g_V(V_N)dV_1\cdots dV_N = (P_{V>V_0})^N$$

Finally,

$$g_{V_{opt}}(V) = N(1 - P_{V \leq V_0})^{N-1} g_V(V)$$

Probabilistic Sampling

We now have a probability distribution characterizing a lower bound on the volume of the optimal intersection volume. Let Q be the event that, for a given match region, a sample point chosen at random within the match region falls in the optimal intersection volume. The probability of Q given V_{opt} is $P(Q|V_{opt}) = \frac{V_{opt}}{M}$ where M is the volume of the match region. Then

$$P_Q = P(Q) = \int_{V=0}^{\infty} g_{V_{opt}} P(Q|V_{opt})dV = \int_{V=0}^{\infty} g_{V_{opt}} \frac{V_{opt}}{M}dV.$$

The probability that no such random sample point falls in the optimal intersection volume is $(1 - P_Q)^N$. The probability that none of k random points per match region falls in the optimal region is $(1 - P_Q)^{kN}$. So, the probability that none of the sample points falls in the optimal intersection volume can be made arbitrarily small by increasing k.

3.6 Hypothesis Refinement

The hypothesis generation step constructs feasible matchings by searching **TPS** for optimal intersection volumes. Because a particular matching may be feasible over a wide range of transformations, some transformations will not align the model as well as others, and a means of determining the optimal T is needed. In the case of uniform sampling, one approach is to pick the sample point, from among those in the optimal intersection volume, which minimizes some error metric such as a simple least sum squares error function of the distance between matched features. Then, starting at this point, a second stage of finer sampling could be done to improve the guess.

In all three approaches to hypothesis generation, a parallel gradient-based search technique could also be used to optimize $F(T) = \sum_{<m_i,d_j>\in \mathbf{M}} f(< m_i, d_j >, T)$ using a definition of $f(< m_i, d_j >, T)$ which yields a smooth function $F(T)$.

Chapter 4

Transformation Sampling Algorithms and Implementations

Chapter 3 introduced the idea of transformation sampling as a basis for 2D model-based recognition. Three methods were introduced: critical point sampling, probabilistic sampling, and uniform sampling. A formal analysis of these was given along with a sketch of a transformation sampling algorithm based on each of them. This chapter outlines in detail algorithms for these techniques, and describes an implementation of uniform sampling. First, a particular model of parallel computation is introduced which facilitates the description and analysis of the algorithms.

4.1 The Vector Model of Parallel Computation

For parallel algorithm design we require a model of parallel computation. The model does not precisely represent a specific parallel architecture, but provides a conceptual framework for algorithmic development and analysis. By their nature, many of the parallel operations that I discuss here are described very naturally in the *vector model* of parallel computation[6][7].

A vector is a set of indexed data objects. A vector model of computation is defined in terms of primitive operations on vectors which return vectors as results. Thus, conceptually we can think of the processors of our parallel architecture as being organized as elements of an indexed vector, where a processor can operate on the elements of data vectors with the same index.

Parallel operations can be grouped into two classes: primitive operations and common

routines built from primitive operations. There are three main classes of vector machine primitive operations we will use here: *elementwise* arithmetic, logical, and symbolic operations, *permutation* operations, and *scan* operations. From these a variety of useful higher level operations are constructed.

4.1.1 Primitive Parallel Operations

Elementwise Operations

Consider a set of data vectors $A_0, \ldots A_l$ each of length K. Elementwise operations are simply procedures which take as arguments the elements of the A_k with the same index, $A_0[i], \ldots A_l[i]$, and operate on them to produce some result $R[i]$. Thus the computation at each index is identical except for the data, and takes place in parallel. For example, after examples by Blelloch[7]:

$$\begin{array}{lcl} A_0 & = & [3 \quad 1 \quad 3 \quad 4 \quad 3 \quad 9 \quad 2 \quad -3] \\ A_1 & = & [1 \quad 7 \quad 3 \quad 2 \quad 1 \quad 3 \quad 6 \quad 5] \\ A_0 + A_1 & = & [4 \quad 8 \quad 6 \quad 6 \quad 4 \quad 12 \quad 8 \quad 2] \\ A_0 \times A_1 & = & [3 \quad 7 \quad 9 \quad 8 \quad 3 \quad 27 \quad 12 \quad -15] \end{array}$$

Permutation Operations

The permutation primitive takes a data vector A and an index vector I as inputs. The result is a vector R where $R[I[j]] = A[j]$. There are two varieties of permutations, either the mapping defined by the vector I is one-to-one, that is, the elements of I are unique, or it is many-to-one. I only consider the former type where the mapping is a true permutation. The latter type requires some rules for combining elements mapped to the same index in the result.

A data vector can be interpreted as a multi-dimensional grid, much as a linear segment of memory in a serial computer can be interpreted in row or column major order to be a multi-dimensional array. An important special case for vision is a two-dimensional mesh. In many early vision computations, functions over local regions of the mesh are common. A special case of a permutation is a *grid permutation* where the elements of a data vector, interpreted as a grid, are mapped onto a new grid by a simple shift operation. Since many parallel machines have special hardware support for this type of mesh permutation to make it more efficient than general permutation, it might be considered a separate permutation operation.

Scan Operation

An extremely useful class of parallel operations are the *scan* and *segmented scan* operations[7][6]. These are based on parallel prefix computations which take a binary associative operator $\oplus$, and a vector $[a_0, a_1, ..., a_{n-1}]$ as input, and return the vector $[a_0, (a_0 \oplus a_1), ..., (a_0 \oplus a_1 \oplus ... \oplus a_{n-1})]$ as the result. Examples of the operator $\oplus$ include +, **maximum**, **minimum**, and or. Blelloch[6] has termed the more common scan operations **+-scan**, **max-scan**, **min-scan**, **or-scan**, **and-scan** and **first-scan**. Examples include:

A	=	[5	1	3	4	3	9	2	6]
+-scan(A)	=	[5	6	9	13	16	25	27	33]
max-scan(A)	=	[5	5	5	5	5	9	9	9]
first-scan(A)	=	[5	5	5	5	5	5	5	5]

An important specialization of the scan operations are the segmented scan operations, where disjoint, contiguous segments of contiguous vector elements are considered subvectors. A vector of boolean values specifies the beginning of new segments. For example:

Segment-Flags	=	[T	F	T	F	F	F	T	F]
A	=	[5	1	3	4	3	9	2	6]
	=	[5	1]	[3	4	3	9]	[2	6]
B	=	[1	0]	[2	0	3	1]	[0	1]
+-scan A	=	[5	6]	[3	7	10	19]	[2	8]
max-scan A	=	[5	5]	[3	4	4	9]	[2	6]
first-scan A	=	[5	5]	[3	3	3	3]	[2	2]
permute A, B	=	[1	5]	[4	9	3	3]	[2	6]

The segmented scan operations enable separate parallel operations on data sets represented as vectors.

4.1.2 Common Parallel Routines

Outer Product

The *outer product* is analogous to the Cartesian product of two sets $\{< a_i, b_j >\} = \{a_i\} \times \{b_i\}$. This procedure takes two vectors as input and returns a vector whose elements consist of all possible pairs of elements in the original two vectors. This is an example of a vector operation which returns a result vector of different size than the input vectors. Outer

product is used, for example, to form the vector of all feature matches from the set of image and model features. This procedure is described in more detail in appendix A. For example:

$$\begin{array}{lcl} \text{Model} & = & [m_0 \quad m_1] \\ \text{Image} & = & [d_0 \quad d_1] \\ \textsf{outer-product}(\text{Model,Image}) & = & [m_0,d_0 \quad m_0,d_1 \quad m_1,d_0 \quad m_1,d_1] \end{array}$$

Generalized Histogram

The standard histogram maps a vector of keys, K, to a new vector H, where any two elements of K, k_i and k_j, are mapped to the same element of H iff $k_i = k_j$. This equivalence relation defines a partition of the elements of K. Let A be an arbitrary data vector where the elements of A and K are in one-to-one correspondence, thus $|A| = |K|$. The same mapping of K to H also defines a partition of the elements a_i of A. Let $\bar{a}_i$ represent the equivalence class of element a_i. Further, let $\phi(\bar{a}_i)$ be a function $\phi : \{\bar{a}_i\} \longrightarrow \Re$ on the set $\{a_j\} \in \bar{a}_i$. The generalized histogram takes as inputs K and A. The result vector, G, is defined as follows. $G[i] = \phi(\bar{a}_i)$ where $\bar{a}_i$ is the equivalence class of the element a_i corresponding to element k_i of K. The generalized histogram is described in more detail in appendix A. For example, let $\phi(\bar{a}_i) = |\bar{a}_i|$, $a_j \in \bar{a}_i$, and let $A = K$. Then the generalized histogram is similar to a standard histogram H, except associated with each element of K is the number of elements in its equivalence class, i.e. with the same value.

$$\begin{array}{lcl} \text{A} & = & [5 \quad 1 \quad 3 \quad 4 \quad 3 \quad 9 \quad 5 \quad 5] \\ \text{K} & = & [5 \quad 1 \quad 3 \quad 4 \quad 3 \quad 9 \quad 5 \quad 5] \\ \text{G} & = & [\mathit{3} \quad \mathit{1} \quad \mathit{2} \quad \mathit{1} \quad \mathit{2} \quad \mathit{1} \quad \mathit{3} \quad \mathit{3}] \\ \text{H} & = & [(5,\mathit{3}) \quad (3,\mathit{2}) \quad (9,\mathit{1}) \quad (4,\mathit{1}) \quad (1,\mathit{1})] \end{array}$$

Expand Vector

The **expand-vector**(A,K) function takes a vector A of data objects, and an equal sized vector of integers K, and returns a new vector where each element $A[i]$ is represented in $K[i]$ contiguous vector elements of the result. Note that the new vector has length $\sum K[i]$. The function **expand-vector-index** is a complimentary function which returns a vector of integers assigning a zero-based index to each copy of a particular element $A[i]$ determining its position in the segment of copies. These operations are described in more detail in appendix A. For example:

A	=	[a	b	c	d]			
K	=	[2	3	2	1]			
expand-vector(A,K)	=	[a	a	b	b	b	c	c	d]
expand-vector-index(K)	=	[0	1	0	1	2	0	1	1]

Sorting

Sorting is conveniently decomposed into two operations: **rank**[1], and **permute**. For a vector A, **rank**(A) returns a vector of equal length of unique indices indicating the sorted rank of each element in A. A sort is then accomplished by permuting the appropriate data elements according to the rank vector. Sorting requires $O(\lg N)$ time for a vector of length N in the vector model, and $O(\lg^2 N)$ time on N processors for a typical machine[25][7].

4.2 Transformation Sampling Algorithms

The parallel operations just introduced form the basic algorithmic building blocks from which the algorithms are constructed. This section provides a detailed development and complexity analysis of hypothesis generation algorithms based on transformation sampling.

4.2.1 Uniform Sampling

An Algorithm

The following outlines the basic steps of an algorithm for uniform sampling.

- Extract image features from input image
- Form all matches $< m, d >$ of model and image features
- Compute the range of feasible relative rotations for each match
- Determine the set of rotation sample points in this range of rotations
- For each rotation sample point, compute the range of feasible relative translations
- Determine the set of translation sample points in this range of translations
- For each complete parameter space sample point T thus constructed, compute the value of $F(T) = \sum f(< m, d >, T)$ at the sample point
- Select the transformation sample points with optimal values of $F(T)$

- Determine the object's pose in the image from the optimal transformation sample points

Complexity of the Algorithm

The algorithm for hypothesis generation by uniform sampling consists of two main steps: computing the set of transformation sample points in the match-region of each feature match; and evaluating $F(\mathbf{M}, T) = F(T)$ for each one, and picking the transformation with the optimal values of this function. This forms hypotheses as to the object's pose.

The complexity of the feature matching stage depends on the number of feature matches mn and on the number of transformation sample points, K, per feature match. Let $\delta\phi$ radians be the sampling interval in the rotational dimension, and δt be the sampling interval in each of the translational dimensions. The number of sample points that fall inside a match-region depends roughly on its volume.

I consider point features and line segment features. For both these types of features the number of distinct rotation sample points falling in the match region is given approximately by $r \leq \lceil \frac{2\Theta}{\delta\phi} \rceil$. See figure 4.1. For any given rotation, the number of translation sample points falling in the match-region at that rotation is different for point features and line segment features. In the case of point features, the range of valid translations defines a circle of radius D, and there are approximately $a = \frac{\pi D^2}{\delta t^2}$ translation samples falling in this region. So for any point feature match there are approximately ar sample points falling in its match region and $(mn)(ar)$ total transformation sample points to consider.

In the case of line segment features the range of relative translations depends on D, and also on the difference in length between the matched model and image line segments, because one feature can slide relative to the other. I approximate the region of valid translations by a rectangle with major axis aligned with the image segment, and of length $2D + L_{ij}$ and width $2D$, where L_{ij} denotes the difference in length between the features of match $< m_i, d_j >$. The number of sample points falling in this region is given by wl_{ij} where $w \leq \lceil \frac{2D}{\delta t} \rceil$, and $l_{ij} \leq \lceil \frac{2D+L_{ij}}{\delta t} \rceil$. So for match $< m_i, d_j >$, the number of discrete transformation sample points considered is defined to be $K_{ij} = rwl_{ij}$.

If we represent each transformation sample point as the element of a vector, we require $\sum_{i,j} K_{ij}$ vector elements to represent all the sample points in the case of line segment features, and $(mn)(ar)$ in the case of point features. Note that the number of transformation samples per match depends on the maximum dimension, I, of the image in two ways. First, as was shown in chapter 3, lower bounds on the size of the sampling intervals $\delta\phi$ and δt depend on I. The actual length of a line segment feature is also bounded above by $I\sqrt{2}$, so

$l_{ij} \leq I\sqrt{2}$; and so $K_{ij} \leq rwI\sqrt{2}$ for any match region. In practice however, line segments are much shorter, and an average case value of l is reasonable, taken over many different pairs model and data feature sets. Defining by such an average $K = rwl$ or $K = ar$, the total number of different transformations that need be considered is approximately Kmn.

Procedurally, the algorithm is quite simple. Starting with the vector of feature matches, each match is replicated once for each feasible rotation sample point as shown in figure 4.1. The copies of each match are arranged as a segment of contiguous vector elements. This is simply the expand-vector operation mentioned earlier. From each copy, the lowest feasible rotation sample can be computed, so based on a match-copy's position in the segment, the particular rotation sample associated with each copy can be computed. For each replicated match associated with a particular rotation sample, the model feature is rotated according to the rotation, forming a set of *rotated-matches.* In the following example, for convenience let $m_i d_j$ represents the match $< m_i, d_j >$ and $m_i' d_j$ represents the match after rotation of the model feature.

Matches	=	$[m_0 d_0$	$m_0 d_1$	$m_1 d_0$	$m_1 d_1]$			
Copies	=	$[m_0 d_0$	$m_0 d_0$	$m_0 d_1$	$m_0 d_1$	$m_1 d_0$	$m_1 d_0$	$m_1 d_1$	$m_1 d_1]$
Rotation-Samples	=	$[\phi_{20}$	ϕ_{21}	ϕ_{13}	ϕ_{14}	ϕ_1	ϕ_2	ϕ_{21}	$\phi_{22}]$
Rotated-Matches	=	$[m_0' d_0$	$m_0' d_0$	$m_0' d_1$	$m_0' d_1$	$m_1' d_0$	$m_1' d_0$	$m_1' d_1$	$m_1' d_1]$
Segment-flags	=	[T	F	T	F	T	F	T	F]

The next step is to determine the valid translation sample points for each such rotation sample point. The same type of operation is performed on each of the rotated-matches, replicating each of them once for each valid translation sample point. The associated translation sample for each copy is determined and each copy is translated accordingly. The result at this point is that each original feature match is replicated once for each feasible transformation sample point, and for each copy the model feature has been transformed according to its associated transformation sample point. In the following example, for convenience $m_i d_j$ represents the match $< m_i d_j >$, and $m_i'' d_j$ represents the match after transformation of the model feature. This shows the result of replicating each feature match once for each feasible transformation sample point, represented by the T_k.

Transformed-matches	=	$[m_0'' d_0$	$m_0'' d_0$	$m_0'' d_0$	$m_9'' d_{21}$	$m_9'' d_{21}$	$m_9'' d_{21}$	$m_7'' d_3$	$m_7'' d_3$	$m_7'' d_3]$
Segment-flags	=	[T	F	F	T	F	F	T	F	F]
Segment-flags	=	$[T_{34}$	T_5	T_{57}	T_{10}	T_{34}	T_{16}	T_{19}	T_5	$T_{34}]$

The final step is to organize the transformed matches according to the transformation applied to them, and compute $F(T)$ over each set of matches with the same transformation

sample point T_k. This is done by sorting the matches according to the T_k, into contiguous vector segments. At this point the function $f(< m, d >, T)$ is computed for each match, and $F(T)$ is computed by scan operations over each segment. For example:

Transformed-matches	=	$[m_0'' d_0$	$m_7'' d_3$	$m_9'' d_{21}$	$m_9'' d_{21}$	$m_7'' d_3$	$m_0'' d_0$	$m_7'' d_3$	$m_9'' d_{21}$	$m_0'' d_0]$
Sample-points	=	$[T_5$	T_5	T_{10}	T_{16}	T_{19}	T_{34}	T_{34}	T_{34}	$T_{57}]$
Segment-flags	=	[T	F	T	T	T	T	F	F	T]

$f(< m, d >, T)$	=	$[f(m_0'' d_0, T_5)$	$f(m_7'' d_3, T_5)$	$f(m_9'' d_{21}, T_{10})$	$\ldots]$
$F(T)$	=	$[F(T_5)$	*	$F(T_{10})$	$\ldots]$

The algorithm described in terms of vector primitives and routines is as follows. Most of the functions in this pseudo code program are fairly obvious in function; they are all described below.

;;; Copy feature matches once for each rotation sample point

Feature-Matches = outer-product(Model-Features,Image-Features)
Rotation-Count = rotation-samples-per-region(Feature-Matches,Θ)
Expanded-Matches-1 = expand-vector(Feature-Matches,Rotation-Count)
Segment-Index-1 = expand-vector-index(Rotation-Count)
Rotation-Sample-Points = segment-index->rotation-index(Expanded-Matches-1,Segment-Index-1)
Rotated-Matches = rotate-model-feature(Expanded-Matches-1, Rotation-Sample-Points)

;;; Copy rotated feature matches once for each translation sample point

Translation-Count = translation-samples-per-rotated-match(Rotated-Matches,D)
Expanded-Matches-2 = expand-vector(Rotated-Matches,Translation-Count)
Segment-index-2 = expand-vector-index(Translation-Count)
Expanded-Rotation-Sample-Points = expand-vector(Rotation-Sample-Point,Translation-Count)

;;; Select the optimal transformation sample point by computing $F(T)$

Translation-Sample-Point = segment-index->translation-index(Expanded-Matches-2,Segment-Index-2)
Transformed-Matches = translate-model-feature(Expanded-Matches,Translation-Sample-Point)
Sample-Point = transformation-sample-point (Translation-Sample-Points, Expanded-Rotation-Sample-Points)
Permutation-Index = rank(Sample-Point)
Transformed-Matches = permute(Transformed-Matches, Permutation-Index)
Sample-Point = permute(Translation-Sample-Points, Permutation-Index)
Segment-Marker = segment(Transformation-Point)
f = f(Transformed-Matches)
F = segmented-F(f, Segment-Marker)

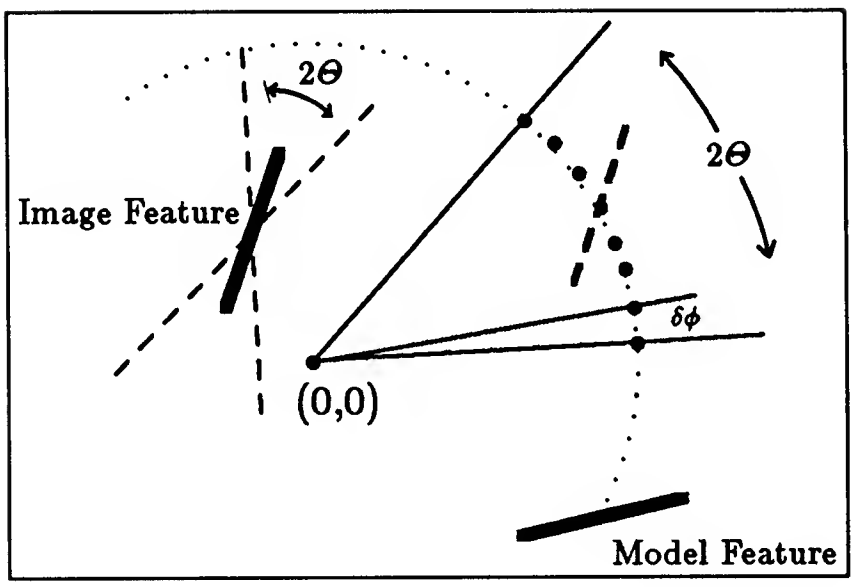

Figure 4.1: An illustration of the uncertainty bound Θ, the sampling interval $\delta\phi$, and the set of possible rotation samples for a particular match.

The **outer-product**, **expand-vector**, and **expand-vector-index** functions have already been described, and require permutation and scan primitive operations. The **segment**, **rank**, and **permute** operations have also been described. The **rotation-samples-per-match** function is an elementwise function calculating the number of rotation samples falling in the range of feasible relative rotations for each feature match. **Segment-index->rotation-index** is an elementwise operation computing the actual rotation sample point associated with a given copy of a feature match depending on the index assigned by **expand-vector-index** in the copying operation. The **rotate-model-feature** function is an elementwise operation computing a new feature match where the model feature has been rotated according to the specified rotation sample point. The functions **translation-samples-per-match**, **segment-index->translation-index**, and **translate-model-feature** are analogous to the above operations for rotations. The function **transformation-sample-point** simply combines the rotation and translation sample points into a complete transformation sample point. Finally, $f(< m, d >, T)$ computes, elementwise, the distance metric on each transformed feature match, and this result is combined for each set of match copies associated with the same transformation sample point by computing $F(T)$ over the results of $f(< m, d >, T)$ for each such set. The sets of matches with the same transformation sample point are arranged in a contiguous segment of vector elements, and $F(T)$ is assumed to be computable by scan operations; thus the last few operation perform a generalized histogram. The histogram keys are the transformation sample points, and the accumulated values are the results of $f(< m, d >, T)$ for matches

at that sample point.

To determine the complexity of this uniform sampling algorithm we must simply determine the complexity of each of the above steps. In the vector model, scan and permute operations take $O(1)$ time, however sorting takes $O(\lg N)$ for a vector of length N, and $O(\lg^2 N)$ time on a typical N processor machine.

. Thus, because there are Kmn transformation sample points considered, the above algorithm requires Kmn processors and $O(\lg^2 Kmn)$ time.

4.2.2 Probabilistic Sampling

The probabilistic sampling approach is in many ways the simplest of the three methods for transformation sampling. The algorithm starts by forming all feature matches. For each match, k random sample points, T, are generated such that they fall anywhere within the match region, according to some predetermined distribution. For each of these kmn sample points, the set of match regions containing it is determined, and $f(< m, d >, T)$ is computed for each match and each random sample point. This corresponds to computing the piecewise constant definition of the metric $F(T)$ for each random sample point T. These sample points are ranked according to the value of $F(T)$, and the best are taken as initial matching and pose hypotheses. We next consider in detail this hypothesis generation process. The following is a sub-optimal but simple algorithm for probabilistic sampling.

We begin with a vector S of all model and image feature matches. The first step is to associate with each feature match $< m, d >$ a copy of the entire set of matches S. This is done by creating a vector, P, where each feature match $< m_i, d_j >$ is the first and *principle* match in a contiguous segment of vector elements. The remainder of the segment consists of a copy of the vector S. There is such a segment with each feature match as principle match, forming the vector P. Construction of this vector is done by permutation and scan operations. For example, let m_i, d_j represent the match $< m_i, d_j >$:

$$
\begin{array}{lcl}
\text{S} & = & [m_0,d_0 \quad m_0,d_1 \quad m_1,d_0 \quad m_1,d_1] \\
\text{P} & = & [m_0,d_0 \quad m_0,d_0 \quad m_0,d_1 \quad m_1,d_0 \quad m_1,d_1 \\
 & & \ m_0,d_1 \quad m_0,d_0 \quad m_0,d_1 \quad m_1,d_0 \quad m_1,d_1 \\
 & & \ m_1,d_0 \quad m_0,d_0 \quad m_0,d_1 \quad m_1,d_0 \quad m_1,d_1 \\
 & & \ m_1,d_1 \quad m_0,d_0 \quad m_0,d_1 \quad m_1,d_0 \quad m_1,d_1] \\
\text{Principle-match} & = & [\mathsf{T} \quad \mathsf{F} \quad \mathsf{F} \quad \mathsf{F} \quad \mathsf{F} \\
 & & \ \mathsf{T} \quad \mathsf{F} \quad \mathsf{F} \quad \mathsf{F} \quad \mathsf{F} \\
 & & \ \mathsf{T} \quad \mathsf{F} \quad \mathsf{F} \quad \mathsf{F} \quad \mathsf{F} \\
 & & \ \mathsf{T} \quad \mathsf{F} \quad \mathsf{F} \quad \mathsf{F} \quad \mathsf{F}]
\end{array}
$$

Each principle processor generates a random sample point, T, and scans a copy out to the remainder of the segment. Each secondary match in the segment computes the value of $f(< m, d >, T)$ for the random sample point, and the result $F(T)$ is accumulated by scan operations back in the principle processor. This procedure is repeated k times, forming kmn transformation hypotheses, ranked by the value of $F(T)$ for each one. This forms an entire set of hypotheses.

To determine the complexity of this approach we simply examine the complexity of the component routines. The algorithm uses $O(m^2n^2)$ processors mn for principle processors and m^2n^2 for copies of S. Because all the operations consist of permutation and scan operations, the algorithm requires time $O(1)$ in the vector model, and $O(\lg kmn)$ time on a typical machine.

A more careful approach would be to use a Hough transform-like procedure to hash the locations of the match regions in **TPS**. In this way, a random sample point finds the match regions which hash to the same place, and in practice much less that m^2n^2 processors are needed.

4.2.3 Critical Point Sampling

Critical point sampling is the most complex of the three techniques, however it is guaranteed to find an optimal feasible matching. Similar to the probabilistic algorithm, the critical point algorithm first computes transformation sample points, T, and then simply computes the function $F(T)$ for each of these. The sample points are constructed by considering the $O(m^3n^3)$ critical points described in chapter 3. A simple procedure for computing $F(T)$ is to use the same technique as in the case of probabilistic sampling, organizing each sample point as a principle element in a segment consisting of all matches. The functions $f(< m, d >, T)$ and $F(T)$ are computed using elementwise and scan computations as before. Because

there are $O(m^3n^3)$ transformation samples, this requires $O(m^4n^4)$ processing elements. Alternatively a hashing scheme as mentioned in the previous section could be used.

The computation of critical points can be done in parallel using m^3n^3 processing elements. This involves finding the roots of 6^{th} degree polynomials. There are numerous numerical methods for finding the roots of polynomials[29]. Forming all triples of matches requires permutations and scans, and can be performed in $O(1)$ time in the vector model or $O(\lg mn)$ time on typical parallel machines. Computing the critical points is an elementwise computation independent of m or n. One subtlety of the approach is that the critical points all lie on the boundaries of the match regions from which they were constructed. Considering the critical points computed from two or three correct match regions, it is likely that the critical points are contained in the interior of the other correct match regions, although it is possible that they also lie in the boundary of some of them. $F(T)$ is simply computed using any matches with match regions containing a critical point in its interior or bounding surface.

So, we see that the sample points with an optimal value of $F(T)$ can be computed in $O(\lg mn)$ time using $O(m^4n^4)$ processing elements.

4.3 Implementation

4.3.1 The Parallel Nature of This Formulation

In the ideal case, a computation running in time $O(f(n))$ on a single processor can be done in time $O(f(n)/N)$ on an $O(N)$ processor machine. The key is to divide up the computation into pieces on which each processor can operate in parallel. In practice this is not always so easy, since at the very least it is often necessary to combine the results of the computations, and possibly information must be exchanged between the processors during the individual computations. This adds to the complexity of the procedure, reducing the optimal $O(N)$ speedup.

Transformation sampling is highly parallel in nature. The basic data elements used in all of the computations are model and image features and feature matches. The operations which we perform on these data are of two basic classes: local operations on the feature or match whose results are only pertinent to the particular feature or match, and global operations where information is accumulated from or distributed over a number of data elements. The parts of the algorithm requiring local computations allow an optimal speedup over a single processor. These include the operations on features and feature matches such

as feature transformation. This type of operation on the set of feature matches requires $O(mn)$ time for a single processor, or $O(1)$ time on mn processors.

The required computations of a non-local nature are limited to reorganizing the data elements, and relatively simple accumulation and distribution operations. These operations are sorting of transformation sample points which can be done in $O(\lg^2 Kmn)$ on a typical machine, for Kmn transformation sample points, and accumulating and distributing results of local computations which is done in $O(1)$ time in the vector model, and typically $O(\lg Kmn)$ time on a typical machine using scan operations.

Scan Computations

One of the main reasons for the parallel nature of these algorithms is that most of the non-local computations can be computed with scan operations. Scan-based computations are extremely fast, requiring $O(1)$ time in the vector model and $O(\lg N)$ time on typical parallel machines, for N elements.

In particular, I have defined in general terms the metric $F(\mathbf{M}, T)$ used to evaluate hypotheses. The interesting feature of this computation is that it involves data that are spread out over many processing elements, and is not simply an elementwise computation. A broad class of functions $F(\mathbf{M}, T)$, however, can be computed by using scan operations. An example is the particular definition of $F(T)$ used here, where

$$F(T) = \sum_{i,j} f(< m_i, d_j >, T)$$

and $f(< m_i, d_j >, T)$ is an elementwise computation. Exploiting the speed of scan operations is key to the parallel nature of the formulation.

Optimal Speedup

It is interesting to consider the speedup achieved by the parallel implementation of the transformation sampling algorithms considered here. One way to do this is to look at the complexity of an algorithm to do the same thing on a single processor machine. Consider a serial algorithm for uniform transformation sampling. For each of mn features matches, we generate K uniform transformation samples. Because there are mn matches, this requires time $O(Kmn)$. For each distinct sample point, we want to compute the value of $F(T)$. If this is an associative combination of $f(< m, d >, T)$, then the value of $F(T)$ can be accumulated as each new sample point is constructed by organizing the sample points in

a balanced tree structure. As the place of each new sample point is found in the tree, the value of $f(< m, d >, T)$ is combined with the partial result of $F(T)$ there. If there are $O(Kmn)$ elements in the tree, each tree operation takes $O(\lg Kmn)$ time, and computing $F(T)$ for all sample points requires $O(Kmn \lg Kmn)$ time. The resulting sample points can be sorted by the value of $F(T)$ in $O(Kmn \lg Kmn)$ time, yielding the best hypotheses.

Determining the best hypothesis thus requires $O(Kmn \lg Kmn)$ time on a single processor. In contrast, the parallel algorithm using Kmn processors requires takes $O(\lg^2 Kmn)$ time, and thus the speedup is a factor $O(\lg Kmn)$ slower than optimal. It is interesting to note, however, that special parallel sorting networks exist that can sort N items in $O(\lg N)$ time, and thus in principle this algorithm offers optimal speedup over a single processor implementation[25].

In the case of a serial algorithm for probabilistic sampling, each of kmn random sample points is compared with mn match regions to compute $F(T)$ for each of the points. This takes $O(km^2n^2)$ time on a single processor. The results can be organized by sorting by $F(T)$, requiring $O(kmn \lg kmn)$ time, for total complexity of $O(km^2n^2)$. The parallel algorithm achieves a speedup a factor of $O(\lg kmn)$ slower than optimal. The procedure for critical point sampling is exactly similar except $O(m^4n^4)$ processors are required in the simple algorithm and again the speedup is a factor $O(\lg kmn)$ slower than optimal.

4.3.2 Data Representation

There are basically two data types to be represented: features and transformation sample points. A point feature is represented by five values: the x and y coordinates of its position, the two components of the unit vector characterizing its orientation, and a number representing the object or image data set to which it belongs.

I use lisp-like structure for representing a feature in each processing element:

```
(def-pvarstruct (point-feature)

  (position-x :signed)
  (position-y :signed)
  (normal-x :signed)
  (normal-y :signed)
  (number :field))
```

A line segment feature is represented in an exactly similar fashion:

```
(def-pvarstruct (feature)

  (center-x :signed)
  (center-y :signed)
  (length :field)
  (normal-x :signed)
  (normal-y :signed)
  (number :field))
```

A transformation sample point consists of the sample index for the u and v components of the transformation, and the sample index for the rotation ϕ. The product of a sample index with the appropriate sampling interval gives the actual transformation (ϕ, u, v) represented. A unique number is associated with each sample point–feature match combination depending on the model from which the model feature came. This keeps the computation for different models orthogonal.

```
(def-pvarstruct (transformation-sample-point)

  (u-index :signed)
  (v-index :signed)
  (phi-index :signed)
  (model :field))
```

These form the basic data elements. The model and image features consist respectively of vectors of model features, m and data features d.

4.3.3 The Connection Machine Implementation

The Connection Machine is a *data parallel*[18][2] computing system based on a parallel processing unit of 64K, 32K, or 16K processors, interfaced with a front end computer such as a Symbolics 3600 series lisp machine. The work in this thesis was done on a 16K processor CM-1. The processors are logically organized as a 16 dimensional hypercube network, are capable of general computation, and each has 4K bits of local memory. The CM-2 has 64K bits of memory per processor as well as hardware support for floating point operations.

The microcode supports the idea of virtual processors, in which each processor simulates two or more processors, dividing the available memory equally. In practice, a 16K CM-1 can simulate a 64K machine for the implementation considered here.

The Connection Machine is programmed through language extensions to the language used on the front end computer, and program execution is controlled by the front end computer. The Connection Machine can be programmed in C* or *Lisp[1], extensions to the C and Common Lisp languages respectively. The work in this thesis was based on a *Lisp implementation. *Lisp is an extension to Common Lisp, allowing programming of the Connection Machine processors, with a Lisp-like syntax, although it does not actually implement Lisp operations. On the CM-1, data is loaded into the Connection Machine through the front end computer, and all data are initially stored on the front end computer.

Images are obtained from a CCD camera using a frame buffer attached directly to the Symbolics front end machine, thus the bandwidth of image i/o transfers is limited to that of the control link between the front end and the Connection Machine.

4.4 The Recognition System

Chapter 2 defined the recognition process as the hypothesis of a feature matching and a transformation, followed by the verification of this hypothesis. Transformation sampling was proposed as a method of generating matching and transformation hypotheses. This section describes in some detail the recognition system implemented on the Connection Machine, based on uniform transformation sampling.

Acquiring Models

Models are built using exactly the same feature extraction procedure as for image data. Building a model simply consists of running the feature extraction portion of the recognition algorithm on the image of an isolated object. In principle, the model could be built more carefully using a higher resolution image and multiple views to reduce the effect of inaccuracies in imaging. This has the advantage that any mismatch between the object in the input image and the stored model could be attributed to the image features. In fact this does not appear to be necessary in practice. When acquired, models are stored as arrays of the components of the features on the front end computer, and are loaded into Connection Machine memory prior to the recognition procedure.

---

[1]Pronounced 'Starlisp'

Feature Extraction

As mentioned, the feature extraction process is identical for the formation of the model and the input data features. The first phase of feature extraction consists of the extraction of edges from the original input image. The images used in this implementation are $256^2 \times 8$ bit CCD video brightness arrays. The edge detector developed by Canny[10][27] is used for edge detection.

The result of edge extraction is a discrete bit map registered with the pixels of the original image corresponding to edges. This edge representation is processed to reduce the edges to single chains of 8-connected edge pixels where each edge pixel has at most two adjacent edge pixels. Each such edge segment is approximated by a set of straight line segments. The segments are fitted to the curve fragments by a recursive splitting algorithm, where each curve segment is recursively split into smaller segments and fitted to lines until an error bound on the distance from an edge pixel to the approximating straight line is achieved.

Once the original image has been loaded into the Connection Machine, the entire feature extraction process is performed on the Connection Machine. The result is a set of straight line segment features representing the object boundaries in the input image, one feature per processor.

Recognition

Recognition has been structured as hypothesis and verification, where the hypothesis step consists of hypothesis generation and hypothesis refinement. Hypothesis generation produces likely candidate matchings utilizing the special definition of the metric on matches, where $f(< m, d >, T) = c > 0$ inside a match region and 0 outside it. Intersection volumes where many feature matches have feasible transformations are indicated by maximal values of $F(T) = \sum f(< m_i, d_j >, T)$. Hypothesis refinement takes these matchings, and selects transformations T optimizing $F(T)$ using an appropriate different definition of $f(< m, d >, T)$. These two steps produce hypotheses for $\mathbf{M}$ and T. The final step is verification of the hypotheses.

In the implementation described here, only the hypothesis generation step was utilized. Hypothesis refinement was not performed and hypotheses consisted of matchings defined by maxima in the piecewise constant definition of $F(T)$ where

$$F(T) = \sum_{<m_i, d_j> \in \mathbf{M}} f(< m_i, d_j >, T),$$

$f(< m_i, d_j >, T) = c_{ij} > 0$, and c_{ij} is defined to be the length of the image segment for $< m_i, d_j >$. In this way intersection volumes with large values of $F(T)$ correspond to matchings and transformations which explain a large part of the image data in terms of the model.

This hypothesis generation stage was so accurate that it was used as the entire recognition engine. The only verification that is performed is to require that the optimal value of $F(T)$ exceeds a quality threshold. In this case the fraction of the model perimeter accounted for by the matching must exceed a lower bound of say 15% or 20%. No further verification was performed. In a sense, this hypothesis generation method performs verification at the level of model and image features by ensuring that at the chosen transformation, all feature matchings are feasible.

Stepping Outside the Abstraction

The vector model of parallel computation provides a convenient abstraction for developing and analyzing parallel algorithms. Indeed, it makes the description of the recognition algorithm quite simple in terms of vector primitives and simple routines such as permutations, scans, outer product and so on. Unfortunately, at this time, a programming language supporting this type of abstraction is not available on the Connection Machine[2], and we must deal more intimately with the architecture of the Connection Machine, programming in *Lisp and Paris[1].

Of primary concern is the fact that we have at our disposal a fixed number of processing elements (on a 16K CM-1, realistically 64K processors) on which we must represent vectors which often exceed this number in length. This requires breaking the problem into smaller vectors which map directly onto the available processors. Much of the complexity of the current implementation of the algorithm is the result of the need to perform this manipulation.

Pre-Pruning of Parameter Space

Although this method requires $O(Kmn)$ processors, in practice K can be large, and we almost always have more transformation sample points to consider than the 64K available processors. As mentioned above, an implementational solution to this problem in general is to simply divide, if possible, the problem up into smaller pieces and operate on as much of the data at a given time as possible. This is the approach taken here.

[2] An interesting lisp-based parallel programming language has been developed by Sabot[30].

In the case of transformation sampling it is possible to reduce the number of transformation sample points considered without actually computing them. First, part of verification is that no hypothesized matching and transformation will be verified if the fraction of the model perimeter explained by the matching is below a verification threshold. This can be exploited to reduce the number of samples considered. Because there are many incorrect matches, there are many feature matches which define match regions which intersect with very few other match regions, and thus any possible matchings including them will fall below the verification threshold. The idea is that there are many regions of transformation parameter space in which only a few matches fall, and any transformation sample point representing such a region can be ignored.

To implement this, a coarse translational Hough transform is computed for each match, at each of its associated rotation sample points. This is much faster than full translation sampling and allows the elimination of unpopular transformations before we consider them further. First, the vector of all feature matches is formed. This vector will most likely fit entirely in the machine since typically $mn < 64K$. After the set of rotation sample points for each match is computed, the matches are sorted by rotation index. These rotation sample points cannot be instantiated at once for all feature matches. Instead they are considered in sorted order. Once all matches associated with a certain rotation sample have been instantiated, a separate translational Hough transform for the matches at each rotation point is computed. This Hough transform is computed such that each match votes for any Hough bin with which its match region intersects, at the given rotation. Any match which falls in a bucket which does not contain matches which together meet the minimum verification threshold are eliminated from further consideration. This Hough transform is done in parallel for many matches at different rotation sample points. If the Hough transform is done carefully, the perimeter accounted for by matches in the bucket is an upper bound for the perimeter accounted for by any matching at a transformation sample point within the bucket. Thus if a rotation sample point and a particular match fall in no bucket above the verification threshold, this rotation need not be considered at the possible translation samples. In practice this procedure considerably reduces the number of transformation sample points that must be explored, as is demonstrated experimentally in chapter 5.

Chapter 5

Experiments

Chapter 3 introduced the idea of searching **TPS** for regions of transformations feasible for large feature matchings. Three methods were proposed to accomplish this: critical point sampling, probabilistic sampling, and uniform sampling. This chapter presents the results of experiments conducted to explore the effectiveness of two of these methods, probabilistic sampling and uniform sampling. The critical point sampling technique provides insight into the structure of **TPS** and provide theoretical bounds on a provably correct search technique, however the technique has not been implemented. The experimentation for uniform and probabilistic sampling takes two forms: simulations using synthetic data, and experiments with real image data.

5.1 Goals of Experimentation

Of primary interest is determining the effectiveness of the techniques, and the computational resources required by them. It is also important to explore the limitation of a technique by determining in what cases it fails, and why it fails. In the case of uniform sampling, of primary interest are the sampling intervals required for acceptable recognition performance, and their effect on the complexity of the algorithm. Recall that Kmn processors are required for this procedure where K is the number of transformation sample points per feature match. Asymptotically, the complexity in terms of the input data set and the model is $O(mn)$. To be useful in practice, however, K must be of reasonable size, and this is one of the things we must consider. In the case of probabilistic sampling, the main issue is whether the optimal intersection volume can be found with high probability by taking k random sample points per match region, and what value of k is required for

accurate recognition with high probability.

5.2 Simulations with Synthetic Data

In order to carefully control the experimental parameters, experiments were conducted with synthetic data consisting of point features. Data sets were constructed to represent the model of the object, as well as to simulate image data derived from a scene. In order to produce natural structural configurations of the data features, they were constructed by sampling points along the boundary contours of real images of objects. Procedurally, the data are constructed as follows. Starting with an image of scene objects, edge detection is performed to extract object boundary contours. Next, the resulting edges are sampled at regularly spaced points and the position and the contour normal at each sample point form a point feature. Two types of images were considered: images with one isolated, known object, and images with several unknown objects, but none of the known objects. The former is used to construct synthetic models, the latter to construct purely spurious data.

To simulate image features due to the object of interest, the model features are individually randomly perturbed in position and orientation, and then the whole set randomly rotated and translated. The position of each feature is perturbed by generating an error vector $\delta\vec{p}$ of length ρ and orientation α such that the position of the image feature $\mathbf{p}_d$ is given by $\mathbf{p}_d = \delta\vec{p} + \mathbf{p}_m$. The value ρ is chosen from the distribution $g_\rho(\rho) = \frac{2\rho}{D^2}$ for $0 \leq \rho \leq D$ and similarly α is characterized by the distribution $g_\alpha(\alpha) = \frac{1}{2\pi}$ for $0 \leq \alpha \leq 2\pi$. The synthetic deviation θ_e in the orientation of the image feature from the model feature is characterized by $g_{\theta_e}(\theta) = \frac{1}{2\Theta}$ for $-\Theta \leq \theta \leq \Theta$. The random transformation applied to the entire perturbed model set is chosen from a uniform distribution of rotations and translations.

To form a complete simulated image, the perturbed model set just described is combined with the simulated spurious data to form a complete image feature set. Figures 5.1, 5.2, and 5.3 show the synthetic model, perturbed and transformed model, and spurious data sets, respectively.

5.2.1 Uniform Sampling

The first battery of experiments were conducted to determine the effect of the sampling intervals, $\delta\phi$ and δt, on the fraction of the model features correctly matched by uniform sampling. The experiments were conducted on the model and perturbed, transformed

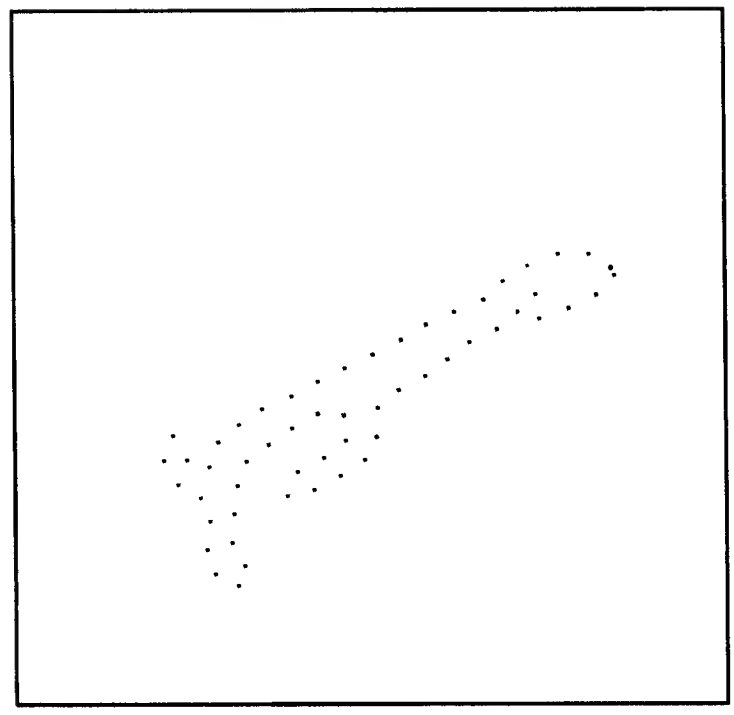

Figure 5.1: Samples of the contour of a real object forming the synthetic model.

model without adding spurious data to test the efficacy of uniform sampling in finding the correct matches.

To minimize the effects of a particular random perturbation or random transformation, for each different combination of sampling intervals, the experiment was run 10 times with a new data set generated each time. The metric $F(T)$ was defined to be the number of image features matched within the uncertainty bounds for each particular transformation sample point T. The sample points were ranked by this value, and the maximal one taken as the result. Figure 5.4 summarizes the results of this experiment. The columns and rows correspond to different rotation and translation sampling intervals respectively. Each table entry displays the average over 10 different runs of the fraction of the model correctly matched, for the sample point with the most feasible matches. The various parameters for data synthesis and the recognition procedure are shown.

The second battery of experiments is similar to the first except that spurious data were also included in the data set along with the randomly perturbed and transformed model. This was to simulate the effects of spurious data arising from unknown objects in the image. The results of these experiments are shown in the tables below. Figure 5.5 displays the fraction of the model correctly matched for the transformation sample point with the largest number of feasible matches. A smaller set of experimental parameters was considered here simply to indicate the general trends. For the sample point with the most feasible matches,

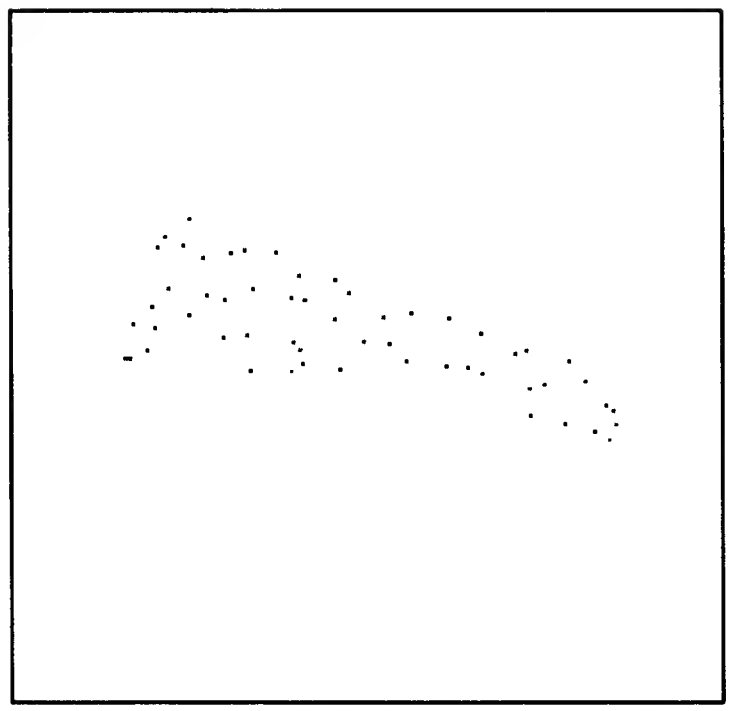

Figure 5.2: The randomly perturbed, randomly transformed model forming the synthetic image features of the object.

figure 5.6 displays the fraction of the matches feasible at the transformation that were actually correctly matched. Figure 5.7 displays the dual of this, the fraction of the feasible matches that were incorrectly matched at the sample point with the most feasible matches.

To demonstrate the effectiveness of the rough Hough transform at pruning out poor sample points before they are considered, figure 5.8 shows the fraction of the total transformation samples that were actually considered after Hough filtering.

5.2.2 Probabilistic Sampling

Experiments were conducted testing the probabilistic sampling approach using synthetic model and image features. The model used was the same as that of the simulations of uniform sampling, and the image features consisted of the same randomly perturbed model without spurious data.

Figure 5.9 displays the results of the experiment, where for each k, $1 \leq k \leq 8$, the average over 10 experiments of the percentage of the features *correctly* matched for the hypothesis with the largest feasible matching is shown.

As a control experiment, an average over 10 runs of a simple alignment technique were done, where instead of a random transformation being chosen, the single, nominal

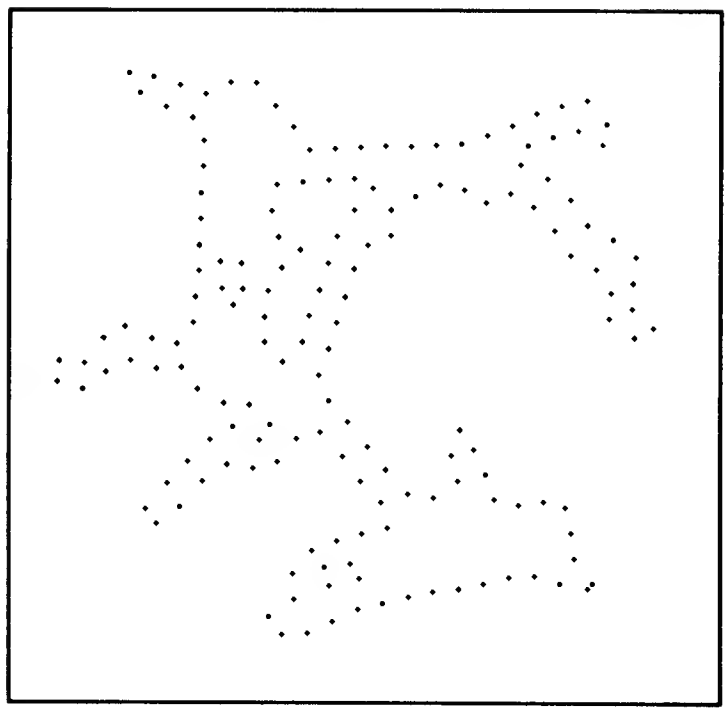

Figure 5.3: Samples of the contour of real, unknown objects, forming the synthetic spurious data.

transformation exactly aligning the measured features was chosen as the transformation sample point for each match region. The average fraction of the model features correctly matched for the hypothesis with the largest feasible matching was 0.74.

5.3 Experiments with Real Image Data

A second class of experiments were conducted using models and objects derived from real images. These experiments were much more qualitative in nature because fewer of the experimental parameters could be controlled.

In the first battery of experiments, different images were considered, each containing the instance of a model as well as many unknown objects partially occluding the known object. For each image, the effects of different sampling intervals were studied as in the synthetic data experiments.

Over the following several pages are displayed the results for several experiments using different data images and three different models. The three models used are shown in figure 5.26. In each case the image and the recognized object are displayed, along with a plot of the extracted image features. The images have been histogram equalized to enhance contrast and reproducibility. In each experiment, the sample points, or hypotheses, were ranked

δt_{pix} \ $\delta\phi_{deg}$	1	2	3	4	5	6	7	8	9	10	11	12
1	0.95	0.93	0.87	0.84	0.82	0.76	0.79	0.74	0.85	0.69	0.75	0.56
2	0.91	0.86	0.90	0.78	0.78	0.78	0.68	0.75	0.71	0.56	0.65	0.61
3	0.85	0.81	0.79	0.76	0.75	0.72	0.77	0.68	0.65	0.62	0.68	0.60
4	0.81	0.82	0.82	0.76	0.76	0.72	0.63	0.71	0.59	0.56	0.52	0.59
5	0.76	0.73	0.69	0.71	0.69	0.68	0.58	0.65	0.49	0.59	0.50	0.48
6	0.72	0.68	0.64	0.62	0.59	0.65	0.58	0.54	0.59	0.51	0.45	0.47
7	0.61	0.67	0.62	0.62	0.57	0.53	0.55	0.53	0.52	0.45	0.47	0.47
8	0.58	0.54	0.55	0.51	0.56	0.50	0.45	0.50	0.47	0.49	0.52	0.45
9	0.5	0.53	0.65	0.55	0.43	0.51	0.48	0.45	0.41	0.51	0.39	0.47
10	0.52	0.53	0.53	0.51	0.48	0.54	0.45	0.47	0.43	0.46	0.43	0.35

Figure 5.4: Average over 10 runs of the fraction of model correctly matched at the sample point with the most feasible matches. There were 56 model features, no spurious data, $D = 5$ pixels, $\Theta = 12$ degrees.

δt_{pix} \ $\delta\phi_{deg}$	1	4	7	10
1	0.94	0.87	0.72	0.60
4	0.82	0.77	0.74	0.57
7	0.61	0.63	0.53	0.51
10	0.59	0.50	0.46	0.45

Figure 5.5: Average fraction of model correctly matched for the sample point with the largest feasible matching, over 10 random trials at each table entry. $m = 56$ model features, and $n = 220$ total data features, including the perturbed model features. $D = 5$ pixels, $\Theta = 12$ degrees.

δt_{pix} \ $\delta \phi_{deg}$	1	4	7	10
1	0.95	0.93	0.95	0.93
4	0.93	0.96	0.95	0.93
7	0.91	0.95	0.88	0.93
10	0.88	0.85	0.86	0.87

Figure 5.6: For the sample point with the most feasible matches, the fraction of the matches feasible that were actually correct, averaged over 10 trials for each table entry. $m = 56$ model features, and $n = 220$ total data features, including the perturbed model features. $D = 5$ pixels, $\Theta = 12$ degrees.

δt_{pix} \ $\delta \phi_{deg}$	1	4	7	10
1	0.05	0.06	0.04	0.04
4	0.06	0.03	0.03	0.04
7	0.06	0.04	0.07	0.04
10	0.07	0.07	0.07	0.06

Figure 5.7: For the sample point with the most feasible matches, the fraction of the matches feasible that were actually incorrect, averaged over 10 trials for each table entry $m = 56$ model features, and $n = 220$ total data features, including the perturbed model features. $D = 5$ pixels, $\Theta = 12$ degrees.

δt_{pix} \ $\delta \phi_{deg}$	1	4	7	10
1	0.03	0.15	0.18	0.4
4	0.04	0.19	0.23	0.44
7	0.07	0.21	0.24	0.45
10	0.09	0.24	0.29	0.46

Figure 5.8: Average fraction of the transformation samples actually considered, after Hough filtering, for 10 random trials at each table entry. $m = 56$ model features, and $n = 220$ total data features, including the perturbed model features. $D = 5$ pixels, $\Theta = 12$ degrees.

k	1	2	3	4	5	6	7	8
	0.74	0.8	0.85	0.88	0.84	0.87	0.88	0.89

Figure 5.9: Each entry is the average over 10 trials, using new perturbed image data each time, of the percentage of the model correctly matched for the sample point with the largest feasible matching.

according to $F(T)$, which was defined to be the sum of the lengths of the image features for matches feasible at T. For each image, three tables summarize the experimental results. The first table for each image displays the rank of the qualitatively different hypothesis that was judged by the author to be correct. Qualitatively different means that, because there are many hypotheses that differ only by a few degrees of rotation and pixels of translation, only one hypothesis from each group of similar hypotheses was considered in the ranking, and this was chosen arbitrarily. A hypothesis was judged correct if the approximately correct transformation had been hypothesized, even if part of the model was not exactly aligned. In these cases, it is likely hypothesis refinement would have improved the approximation to the actual transformation. The first table with each image displays the rank of the hypothesis that was judged correct. The second table displays, for this hypothesis, the percentage of the model perimeter accounted for by the image features, for the hypothesis judged correct. The third table displays the average number of sample points considered per feature match, that is, the quantity $K_{ave} = \frac{\sum K_{ij}}{mn}$.

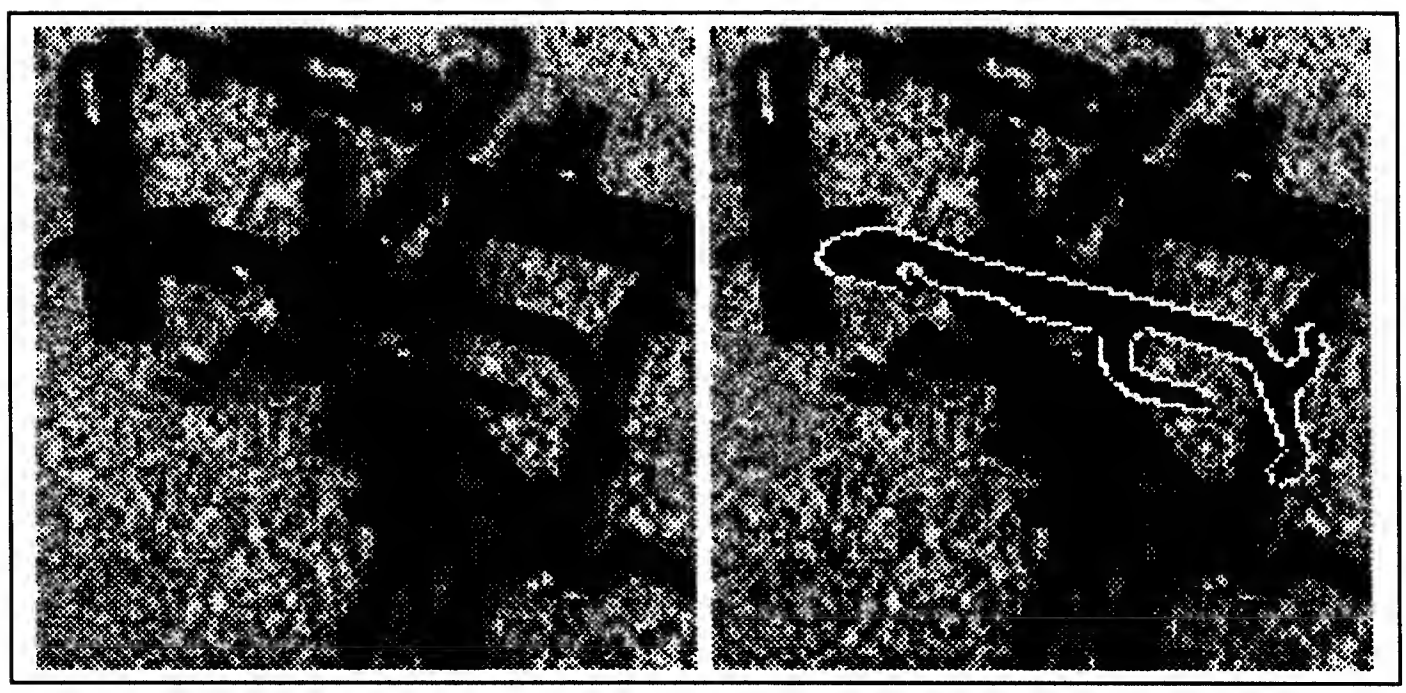

Figure 5.10: The image data, a correct hypothesis, and the image features.

δt_{pix} \ $\delta\phi_{deg}$	2	3	4	5	6	7	8	9	10	11	12
2	1	1	1	1	1	1	1	1	1	1	1
3	1	1	1	1	1	1	1	1	1	1	1
4	1	1	1	1	1	1	1	1	1	1	1
5	1	1	1	1	1	1	1	1	1	1	1
6	1	1	1	1	1	1	1	1	1	1	1
7	1	1	1	1	1	1	1	1	1	1	1
8	1	1	1	1	1	1	1	1	1	1	1
9	1	1	1	1	1	1	1	1	1	1	1
10	1	1	1	1	1	1	1	1	1	1	1

δt_{pix} \ $\delta\phi_{deg}$	2	3	4	5	6	7	8	9	10	11	12
2	0.3	0.32	0.32	0.28	0.32	0.23	0.25	0.3	0.26	0.3	0.32
3	0.32	0.32	0.32	0.27	0.32	0.23	0.25	0.3	0.24	0.3	0.32
4	0.3	0.3	0.3	0.24	0.3	0.23	0.25	0.28	0.24	0.3	0.3
5	0.29	0.29	0.29	0.25	0.29	0.22	0.25	0.27	0.24	0.27	0.29
6	0.32	0.32	0.32	0.23	0.32	0.23	0.25	0.3	0.21	0.3	0.32
7	0.27	0.27	0.27	0.24	0.27	0.23	0.23	0.25	0.24	0.26	0.27
8	0.26	0.26	0.26	0.22	0.26	0.22	0.22	0.26	0.20	0.25	0.26
9	0.23	0.23	0.23	0.21	0.23	0.19	0.19	0.21	0.20	0.23	0.23
10	0.28	0.28	0.28	0.24	0.28	0.19	0.25	0.27	0.24	0.23	0.28

δt_{pix} \ $\delta\phi_{deg}$	2	3	4	5	6	7	8	9	10	11	12
2	204.6	136.3	101.79	81.63	68.28	33.93	34.09	34.22	34.23	33.99	33.94
3	89.76	59.86	44.63	35.8	29.97	14.87	14.95	15.04	15.03	14.93	14.89
4	51.19	34.09	25.43	20.43	17.08	8.48	8.5	8.56	8.55	8.5	8.48
5	31.47	20.96	15.65	12.56	10.5	5.22	5.24	5.27	5.27	5.22	5.23
6	22.46	15.0	11.17	8.97	7.5	3.73	3.73	3.76	3.76	3.74	3.73
7	16.67	11.12	8.3	6.65	5.57	2.77	2.77	2.79	2.78	2.79	2.77
8	12.83	8.54	6.37	5.12	4.28	2.12	2.13	2.15	2.14	2.13	2.12
9	10.13	6.74	5.04	4.05	3.37	1.68	1.68	1.69	1.69	1.69	1.67
10	8.20	5.46	4.08	3.28	2.74	1.37	1.37	1.37	1.37	1.36	1.36

Figure 5.11: Top: The rank, among qualitatively different hypotheses, of the transformation sample point judged correct. Middle: For the transformation sample point judged correct, the fraction of the model perimeter explained by the image features for matches feasible at this sample point. Bottom: For the transformation sample point judged correct, the average number, K_{ave}, of transformation sample points considered per feature match. $m = 41$, $n = 181$, $D = 5$ pixels, and $\Theta = 6$ degrees. The image is shown in figure 5.10.

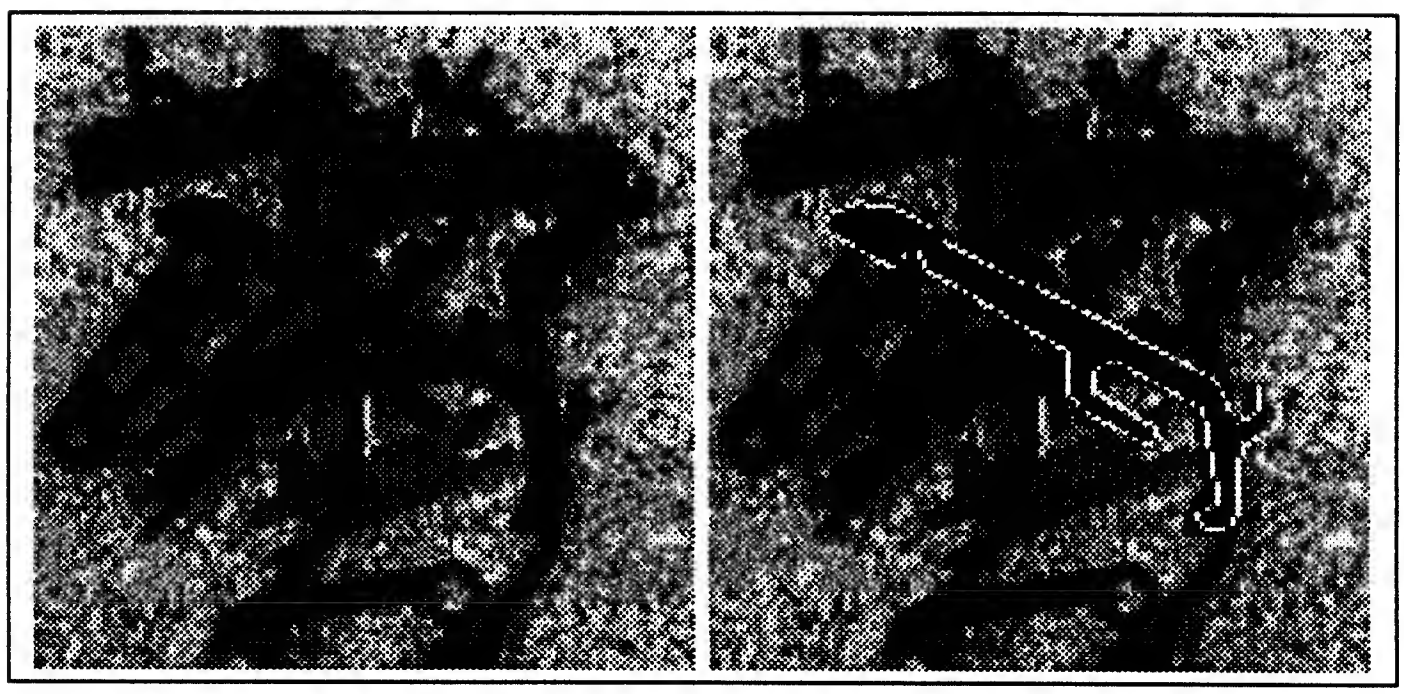

Figure 5.12: The image data, a correct hypothesis, and the image features.

δt_{pix} \ $\delta\phi_{deg}$	2	3	4	5	6	7	8	9	10	11	12
2	1	1	1	1	1	7	1	1	1	1	1
3	1	1	1	1	1	6	1	1	1	1	1
4	1	1	1	1	1	7	1	1	1	1	1
5	1	1	1	1	1	6	2	2	1	1	1
6	1	1	1	1	1	6	3	3	1	2	1
7	1	1	1	1	1	7	4	2	1	1	1
8	1	1	1	1	1	7	1	1	1	1	1
9	1	1	1	1	1	7	4	6	1	4	1
10	1	1	1	1	1	6	3	4	1	5	1

δt_{pix} \ $\delta\phi_{deg}$	2	3	4	5	6	7	8	9	10	11	12
2	0.33	0.33	0.33	0.33	0.33	0.12	0.21	0.21	0.27	0.21	0.33
3	0.3	0.3	0.3	0.3	0.3	0.12	0.19	0.21	0.25	0.21	0.3
4	0.3	0.3	0.3	0.3	0.3	0.11	0.21	0.21	0.25	0.21	0.3
5	0.33	0.33	0.33	0.33	0.33	0.12	0.18	0.18	0.27	0.18	0.33
6	0.28	0.28	0.28	0.28	0.28	0.12	0.15	0.16	0.22	0.16	0.28
7	0.33	0.33	0.33	0.33	0.33	0.11	0.15	0.19	0.27	0.19	0.33
8	0.29	0.29	0.29	0.29	0.29	0.1	0.21	0.21	0.23	0.21	0.29
9	0.21	0.20	0.20	0.20	0.20	0.11	0.14	0.13	0.18	0.13	0.2
10	0.25	0.25	0.25	0.25	0.25	0.12	0.14	0.13	0.19	0.13	0.25

δt_{pix} \ $\delta\phi_{deg}$	2	3	4	5	6	7	8	9	10	11	12
2	239.59	160.4	118.89	95.8	80.29	39.62	39.77	39.97	39.85	39.84	39.65
3	105.09	70.41	52.12	41.99	35.26	17.4	17.47	17.6	17.49	17.49	17.4
4	59.98	40.15	29.74	23.94	20.09	9.92	9.95	10.01	9.96	9.97	9.9
5	36.89	24.69	18.32	14.75	12.37	6.1	6.15	6.16	6.15	6.15	6.11
6	26.3	17.62	13.05	10.52	8.81	4.35	4.37	4.4	4.37	4.37	4.35
7	19.56	13.11	9.7	7.83	6.56	3.25	3.24	3.27	3.25	3.26	3.24
8	15.02	10.06	7.44	6.01	5.03	2.48	2.49	2.5	2.5	2.5	2.47
9	11.85	7.93	5.88	4.73	3.97	1.95	1.97	1.98	1.97	1.97	1.96
10	9.6	6.42	4.77	3.84	3.21	1.58	1.59	1.6	1.6	1.6	1.58

Figure 5.13: Top: The rank, among qualitatively different hypotheses, of the transformation sample point judged correct. Middle: For the transformation sample point judged correct, the fraction of the model perimeter explained by the image features for matches feasible at this sample point. Bottom: For the transformation sample point judged correct, the average number, K_{ave}, of transformation sample points considered per feature match. $m = 35$, $n = 223$, $D = 5$ pixels, and $\Theta = 6$ degrees. The image is shown in figure 5.12.

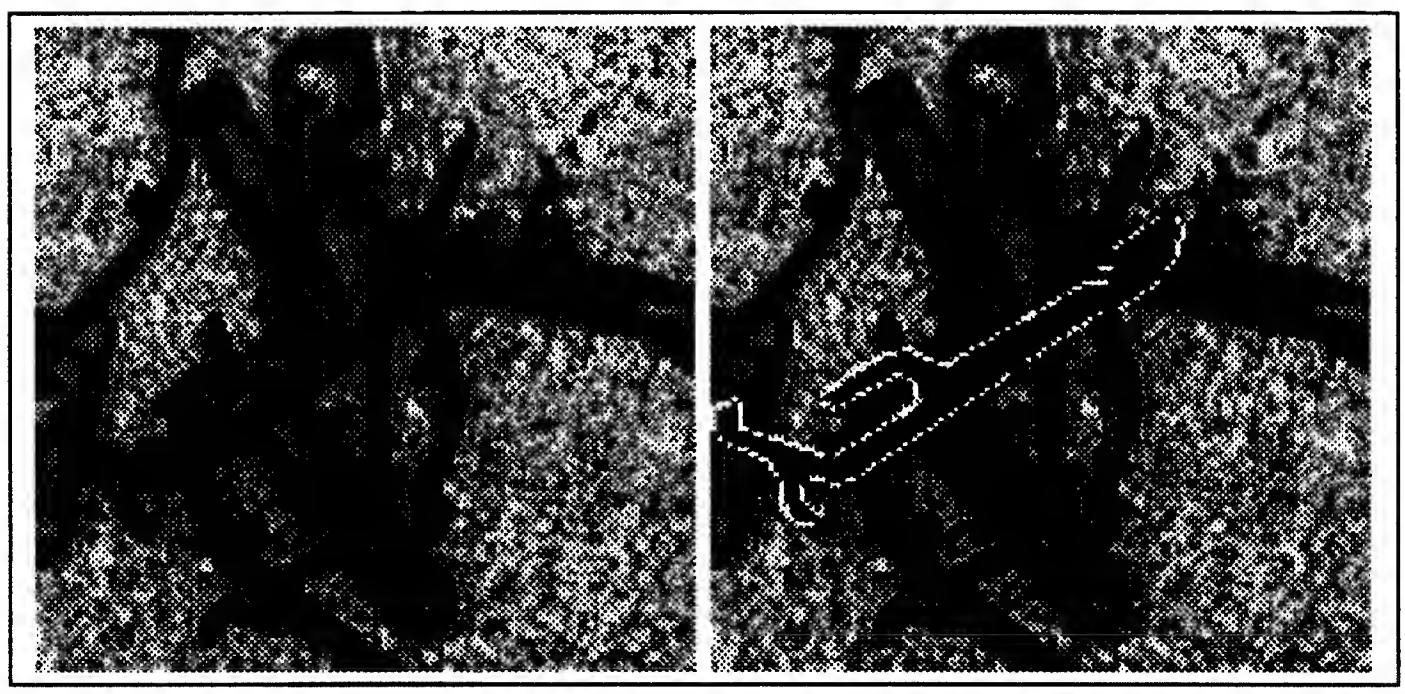

Figure 5.14: The image data, a correct hypothesis, and the image features.

δt_{pix} \ $\delta\phi_{deg}$	2	3	4	5	6	7	8	9	10	11	12
2	11	15	*	*	9	*	*	*	*	*	*
3	15	14	*	*	*	*	*	*	*	*	*
4	14	14	*	*	*	*	*	*	*	*	*
5	15	14	*	*	*	*	*	*	*	*	*
6	12	11	*	*	*	*	*	*	*	*	*
7	15	14	*	*	*	*	*	*	*	8	*
8	*	*	*	*	7	*	*	*	*	*	*
9	11	10	*	*	*	*	*	*	*	*	*
10	13	13	*	*	*	*	*	*	*	*	*

δt_{pix} \ $\delta\phi_{deg}$	2	3	4	5	6	7	8	9	10	11	12
2	0.16	0.12	*	*	0.16	*	*	*	*	*	*
3	0.12	0.12	*	*	*	*	*	*	*	*	*
4	0.12	0.12	*	*	*	*	*	*	*	*	*
5	0.12	0.12	*	*	*	*	*	*	*	*	*
6	0.12	0.12	*	*	*	*	*	*	*	*	*
7	0.12	0.12	*	*	*	*	*	*	*	0.12	*
8	*	*	*	*	0.13	*	*	*	*	*	*
9	0.12	0.12	*	*	*	*	*	*	*	*	*
10	0.12	0.12	*	*	*	*	*	*	*	*	*

δt_{pix} \ $\delta\phi_{deg}$	2	3	4	5	6	7	8	9	10	11	12
2	233.99	156.67	*	*	78.32	*	*	*	*	*	*
3	102.68	68.78	*	*	*	*	*	*	*	*	*
4	58.49	39.18	*	*	*	*	*	*	*	*	*
5	35.99	24.1	*	*	*	*	*	*	*	*	*
6	25.67	17.22	*	*	*	*	*	*	*	*	*
7	19.08	12.78	*	*	*	*	*	*	*	3.17	*
8	*	*	*	*	4.88	*	*	*	*	*	*
9	11.6	7.76	*	*	*	*	*	*	*	*	*
10	9.37	6.27	*	*	*	*	*	*	*	*	*

Figure 5.15: Top: The rank, among qualitatively different hypotheses, of the transformation sample point judged correct. Middle: For the transformation sample point judged correct, the fraction of the model perimeter explained by the image features for matches feasible at this sample point. Bottom: For the transformation sample point judged correct, the average number, K_{ave}, of transformation sample points considered per feature match. $m = 35$, $n = 189$, $D = 5$ pixels, and $\Theta = 6$ degrees. The *'s indicate that the correct hypothesis was not in the top 30 qualitatively different hypotheses, or accounted for less that 7% of the model perimeter. The image is shown in figure 5.14.

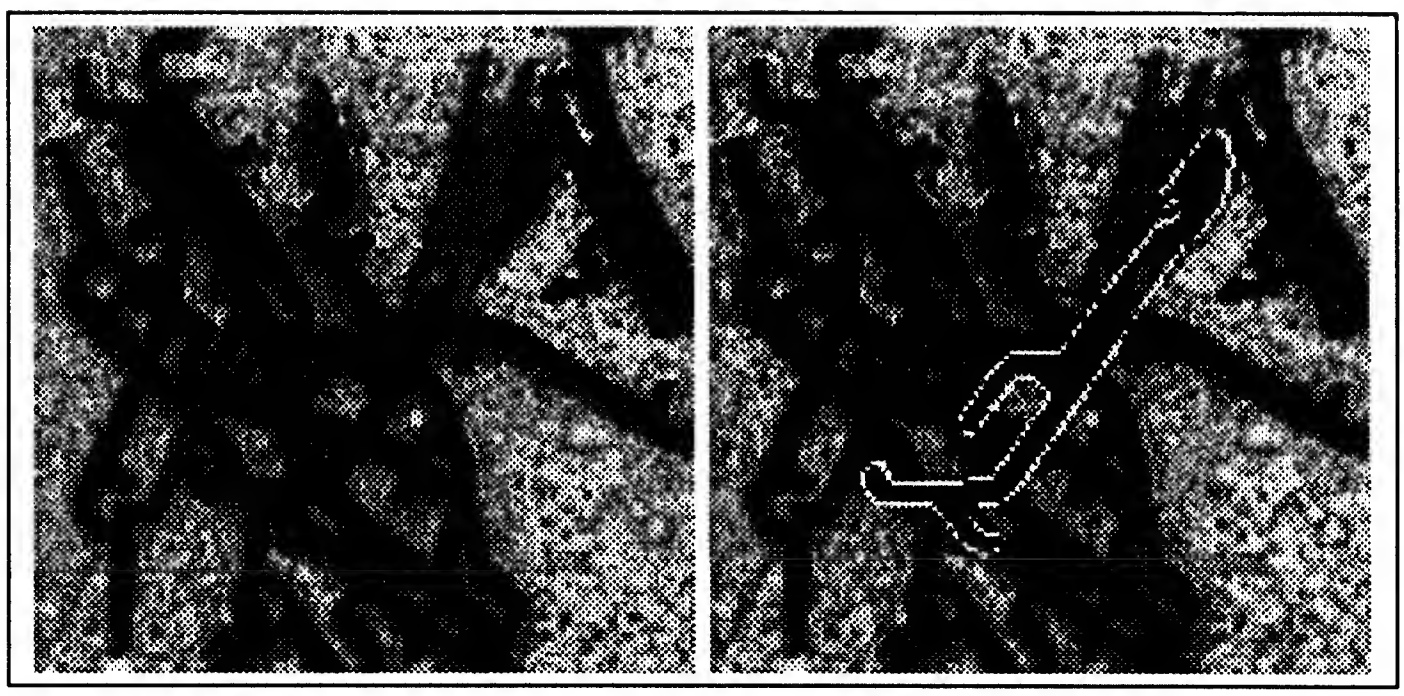

Figure 5.16: The image data, a correct hypothesis, and the image features.

δt_{pix} \ $\delta \phi_{deg}$	2	3	4	5	6	7	8	9	10	11	12
2	1	2	1	1	2	1	*	1	13	2	1
3	2	2	1	1	2	1	*	1	13	2	1
4	2	2	1	1	2	1	*	1	11	2	1
5	2	3	3	1	2	1	*	3	11	2	1
6	2	2	5	1	3	6	*	4	12	1	3
7	1	1	1	1	1	1	*	1	13	*	1
8	1	1	1	2	1	*	*	1	11	1	1
9	4	3	5	3	5	*	*	4	*	5	5
10	3	3	2	4	2	8	*	1	10	6	1

δt_{pix} \ $\delta \phi_{deg}$	2	3	4	5	6	7	8	9	10	11	12
2	0.20	0.20	0.20	0.23	0.20	0.16	*	0.19	0.1	0.18	0.2
3	0.20	0.20	0.20	0.23	0.20	0.16	*	0.19	0.1	0.18	0.2
4	0.20	0.20	0.20	0.21	0.20	0.16	*	0.19	0.1	0.16	0.2
5	0.20	0.18	0.18	0.23	0.18	0.16	*	0.17	0.1	0.18	0.18
6	0.18	0.18	0.15	0.20	0.15	0.13	*	0.14	0.08	0.18	0.15
7	0.20	0.20	0.20	0.22	0.20	0.14	*	0.19	0.07	*	0.2
8	0.19	0.19	0.19	0.16	0.19	*	*	0.18	0.1	0.16	0.19
9	0.16	0.17	0.15	0.16	0.15	*	*	0.14	*	0.13	0.15
10	0.17	0.17	0.17	0.16	0.17	0.09	*	0.16	0.1	0.12	0.17

δt_{pix} \ $\delta \phi_{deg}$	2	3	4	5	6	7	8	9	10	11	12
2	231.52	155.05	114.85	92.58	77.68	38.36	*	38.6	38.58	38.55	38.31
3	101.6	68.09	50.41	40.67	34.08	16.85	*	16.99	16.97	16.91	16.82
4	57.83	38.71	28.69	23.14	19.4	9.56	*	9.63	9.64	9.62	9.59
5	35.7	23.9	17.71	14.27	11.97	5.91	*	5.95	5.95	5.94	5.9
6	25.37	17.01	12.61	10.16	8.52	4.20	*	4.25	4.24	4.24	4.21
7	18.86	12.66	9.35	7.54	6.34	3.13	*	3.16	3.14	*	3.12
8	14.45	9.67	7.16	5.78	4.84	*	*	2.4	2.4	2.4	2.39
9	11.44	7.68	5.68	4.58	3.84	*	*	1.91	*	1.92	1.89
10	9.3	6.22	4.62	3.72	3.12	1.54	*	1.54	1.55	1.55	1.53

Figure 5.17: Top: The rank, among qualitatively different hypotheses, of the transformation sample point judged correct. Middle: For the transformation sample point judged correct, the fraction of the model perimeter explained by the image features for matches feasible at this sample point. Bottom: For the transformation sample point judged correct, the average number, K_{ave}, of transformation sample points considered per feature match. $m = 35$, $n = 234$, $D = 5$ pixels, and $\Theta = 6$ degrees. The *'s indicate that the correct hypothesis was not in the top 10 qualitatively different hypotheses, or accounted for less that 10% of the model perimeter. The image is shown in figure 5.16.

Figure 5.18: The image data, a correct hypothesis, and the image features.

δt_{pix} \ $\delta\phi_{deg}$	2	3	4	5	6	7	8	9	10	11	12
2	1	1	1	1	1	1	1	1	1	1	1
3	1	1	1	1	1	1	1	1	1	1	1
4	1	1	1	1	1	1	1	1	1	1	1
5	1	1	1	1	1	1	1	1	1	1	1
6	1	1	1	1	1	1	1	1	1	1	1
7	1	1	1	1	1	1	1	1	1	1	1
8	1	1	1	1	1	1	1	1	1	1	1
9	1	1	1	1	1	1	1	1	1	1	2
10	1	1	1	1	1	1	2	1	1	1	1

δt_{pix} \ $\delta\phi_{deg}$	2	3	4	5	6	7	8	9	10	11	12
2	0.29	0.28	0.29	0.27	0.27	0.17	0.22	0.23	0.26	0.28	0.25
3	0.29	0.27	0.29	0.28	0.27	0.17	0.20	0.23	0.26	0.27	0.21
4	0.29	0.28	0.29	0.27	0.27	0.17	0.20	0.23	0.25	0.27	0.21
5	0.26	0.26	0.26	0.25	0.25	0.17	0.20	0.21	0.23	0.26	0.24
6	0.29	0.26	0.29	0.24	0.24	0.17	0.20	0.21	0.24	0.27	0.2
7	0.26	0.27	0.26	0.26	0.22	0.17	0.20	0.23	0.24	0.26	0.22
8	0.27	0.27	0.26	0.22	0.27	0.16	0.20	0.21	0.20	0.26	0.2
9	0.29	0.26	0.29	0.24	0.24	0.17	0.20	0.21	0.24	0.27	0.13
10	0.21	0.20	0.21	0.19	0.20	0.16	0.15	0.19	0.19	0.19	0.2

δt_{pix} \ $\delta\phi_{deg}$	2	3	4	5	6	7	8	9	10	11	12
2	213.35	142.04	105.69	84.36	71.21	35.17	35.24	35.3	35.32	35.31	35.17
3	93.85	62.44	46.46	37.1	31.29	15.48	15.5	15.53	15.52	15.54	15.43
4	53.3	35.48	26.38	21.07	17.77	8.78	8.81	8.81	8.82	8.83	8.79
5	32.81	21.86	16.26	12.98	10.94	5.4	5.43	5.44	5.44	5.44	5.4
6	23.46	15.61	11.62	9.25	7.82	3.88	3.86	3.86	3.86	3.89	3.85
7	17.4	11.59	8.62	6.87	5.8	2.87	2.88	2.88	2.87	2.89	2.87
8	13.34	8.87	6.6	5.28	4.45	2.20	2.20	2.21	2.21	2.21	2.2
9	10.57	7.03	5.23	4.18	3.52	1.74	1.74	1.75	1.75	1.75	1.73
10	8.53	5.69	4.23	3.38	2.85	1.4	1.41	1.41	1.42	1.41	1.41

Figure 5.19: Top: The rank, among qualitatively different hypotheses, of the transformation sample point judged correct. Middle: For the transformation sample point judged correct, the fraction of the model perimeter explained by the image features for matches feasible at this sample point. Bottom: For the transformation sample point judged correct, the average number, K_{ave}, of transformation sample points considered per feature match. $m = 43$, $n = 224$, $D = 5$ pixels, and $\Theta = 6$ degrees. The image is shown in figure 5.18.

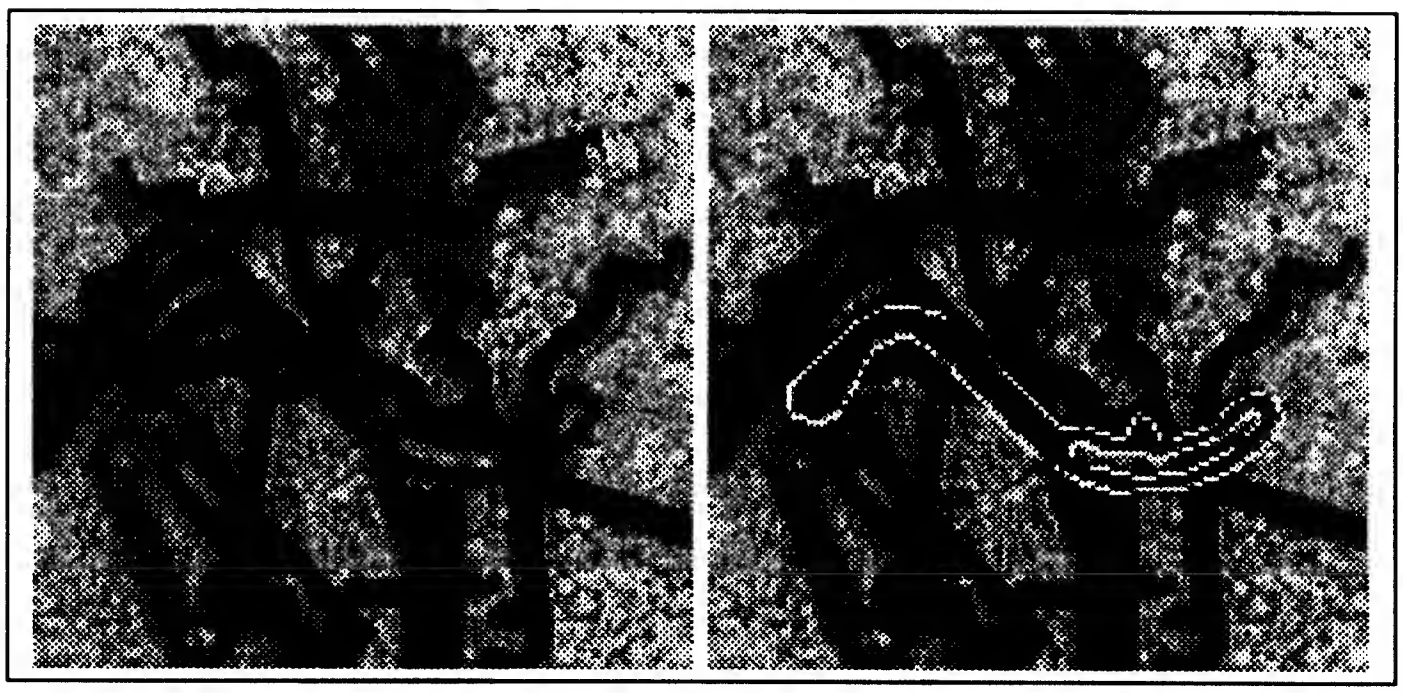

Figure 5.20: The image data, a correct hypothesis, and the image features.

δt_{pix} \ $\delta \phi_{deg}$	2	3	4	5	6	7	8	9	10	11	12
2	2	2	2	1	2	3	*	4	*	1	*
3	2	2	2	1	2	7	5	3	2	1	*
4	1	1	2	3	1	*	*	3	*	1	*
5	3	2	3	3	2	7	6	3	*	1	*
6	2	4	*	5	4	10	*	2	1	2	5
7	1	2	1	1	1	1	5	3	*	1	*
8	4	*	*	*	6	*	*	3	1	3	*
9	*	7	*	7	*	*	*	*	*	*	*
10	1	1	3	3	1	*	6	5	1	1	*

δt_{pix} \ $\delta \phi_{deg}$	2	3	4	5	6	7	8	9	10	11	12
2	0.19	0.19	0.17	0.19	0.19	0.12	*	0.14	*	0.19	*
3	0.19	0.17	0.17	0.18	0.19	0.11	0.11	0.14	0.12	0.19	*
4	0.19	0.19	0.16	0.16	0.19	*	*	0.14	*	0.19	*
5	0.16	0.18	0.15	0.16	0.18	0.1	0.11	0.13	*	0.18	*
6	0.15	0.14	*	0.14	0.14	0.09	*	0.12	0.12	0.14	0.14
7	0.18	0.18	0.17	0.18	0.18	0.12	0.11	0.14	*	0.18	*
8	0.15	*	*	*	0.14	*	*	0.12	0.12	0.14	*
9	*	0.12	*	0.12	*	*	*	*	*	*	*
10	0.18	0.18	0.15	0.16	0.18	*	0.09	0.1	0.13	0.18	*

δt_{pix} \ $\delta \phi_{deg}$	2	3	4	5	6	7	8	9	10	11	12
2	209.58	139.73	103.83	83.02	69.97	34.62	*	34.81	*	34.75	*
3	92.24	61.52	45.66	36.54	30.79	15.25	15.26	15.32	15.29	15.29	*
4	52.37	34.92	25.96	20.75	17.48	*	*	8.69	*	8.69	*
5	32.33	21.55	16.02	12.8	10.8	5.35	5.37	5.38	*	5.37	*
6	23.04	15.37	*	9.11	7.68	3.81	*	3.83	3.81	3.82	3.79
7	17.06	11.37	8.45	6.78	5.69	2.83	2.83	2.83	*	2.83	*
8	13.09	*	*	*	4.37	*	*	2.17	2.16	2.17	*
9	*	6.9	*	4.1	*	*	*	*	*	*	*
10	8.36	5.57	4.14	3.32	2.79	*	1.39	1.39	1.38	1.39	*

Figure 5.21: Top: The rank, among qualitatively different hypotheses, of the transformation sample point judged correct. Middle: For the transformation sample point judged correct, the fraction of the model perimeter explained by the image features for matches feasible at this sample point. Bottom: For the transformation sample point judged correct, the average number, K_{ave}, of transformation sample points considered per feature match. $m = 43$, $n = 237$, $D = 5$ pixels, and $\Theta = 6$ degrees. The *'s indicate that the correct hypothesis was not in the top 10 qualitatively different hypotheses, or accounted for less that 10% of the model perimeter. The image is shown in figure 5.20.

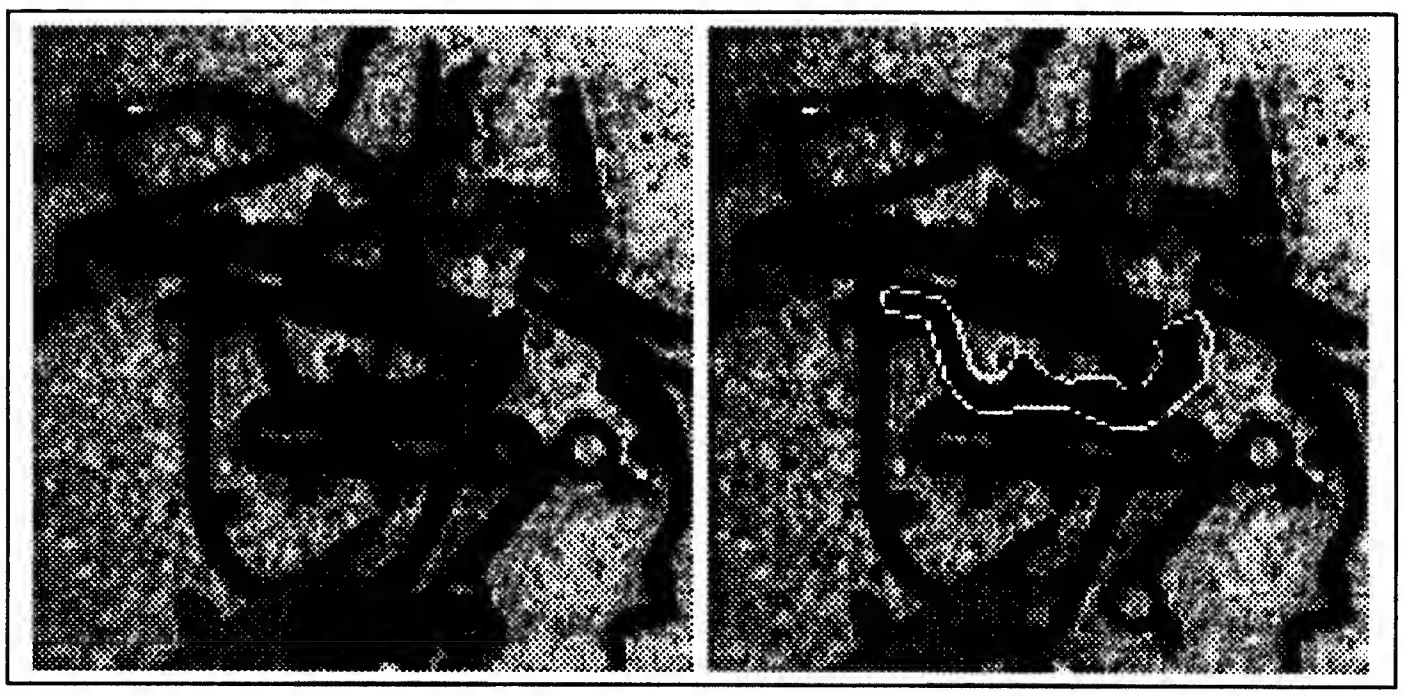

Figure 5.22: The image data, a correct hypothesis, and the image features.

δt_{pix} \ $\delta\phi_{deg}$	2	3	4	5	6	7	8	9	10	11	12
2	1	1	1	1	1	2	1	1	1	1	5
3	1	1	1	1	1	2	1	1	1	1	3
4	1	1	1	1	1	2	1	1	1	1	2
5	1	1	1	1	1	1	1	1	1	1	5
6	1	1	1	1	1	2	2	1	1	2	3
7	1	1	1	1	1	1	1	1	1	1	1
8	1	1	1	1	1	1	2	1	1	1	3
9	2	5	4	*	5	*	4	8	*	5	2
10	7	6	7	*	*	*	2	*	*	7	*

δt_{pix} \ $\delta\phi_{deg}$	2	3	4	5	6	7	8	9	10	11	12
2	0.32	0.3	0.29	0.29	0.29	0.15	0.24	0.29	0.29	0.24	0.18
3	0.29	0.29	0.29	0.29	0.29	0.15	0.24	0.29	0.29	0.24	0.18
4	0.32	0.29	0.29	0.29	0.29	0.14	0.24	0.29	0.29	0.24	0.18
5	0.29	0.29	0.29	0.29	0.29	0.14	0.20	0.29	0.29	0.20	0.17
6	0.29	0.29	0.29	0.29	0.29	0.14	0.18	0.29	0.29	0.18	0.18
7	0.29	0.29	0.26	0.29	0.29	0.14	0.24	0.29	0.29	0.24	0.18
8	0.29	0.29	0.23	0.29	0.29	0.14	0.19	0.29	0.29	0.19	0.16
9	0.19	0.16	0.16	*	0.16	*	0.13	0.12	*	0.13	0.16
10	0.15	0.15	0.15	*	*	*	0.13	*	*	0.13	*

δt_{pix} \ $\delta\phi_{deg}$	2	3	4	5	6	7	8	9	10	11	12
2	190.81	127.72	94.59	76.37	64.14	31.49	31.6	31.6	31.68	31.68	31.6
3	84.28	56.41	41.8	33.73	28.35	13.92	13.97	13.95	13.99	13.98	13.96
4	47.74	31.98	23.67	19.09	16.08	7.87	7.9	7.91	7.94	7.96	7.92
5	29.58	19.8	14.66	11.82	9.95	4.89	4.9	4.92	4.92	4.91	4.9
6	21.02	14.06	10.42	8.41	7.07	3.46	3.48	3.48	3.49	3.5	3.48
7	15.54	10.41	7.71	6.23	5.23	2.56	2.58	2.57	2.58	2.58	2.58
8	11.91	7.99	5.9	4.77	4.0	1.97	1.98	1.97	1.98	1.99	1.97
9	9.41	6.3	4.66	*	3.16	*	1.56	1.55	*	1.56	1.55
10	7.64	5.11	3.79	*	*	*	1.27	*	*	1.27	*

Figure 5.23: Top: The rank, among qualitatively different hypotheses, of the transformation sample point judged correct. Middle: For the transformation sample point judged correct, the fraction of the model perimeter explained by the image features for matches feasible at this sample point. Bottom: For the transformation sample point judged correct, the average number, K_{ave}, of transformation sample points considered per feature match. $m = 27$, $n = 258$, $D = 5$ pixels, and $\Theta = 6$ degrees. The *'s indicate that the correct hypothesis was not in the top 10 qualitatively different hypotheses, or accounted for less that 10% of the model perimeter. The image is shown in figure 5.22.

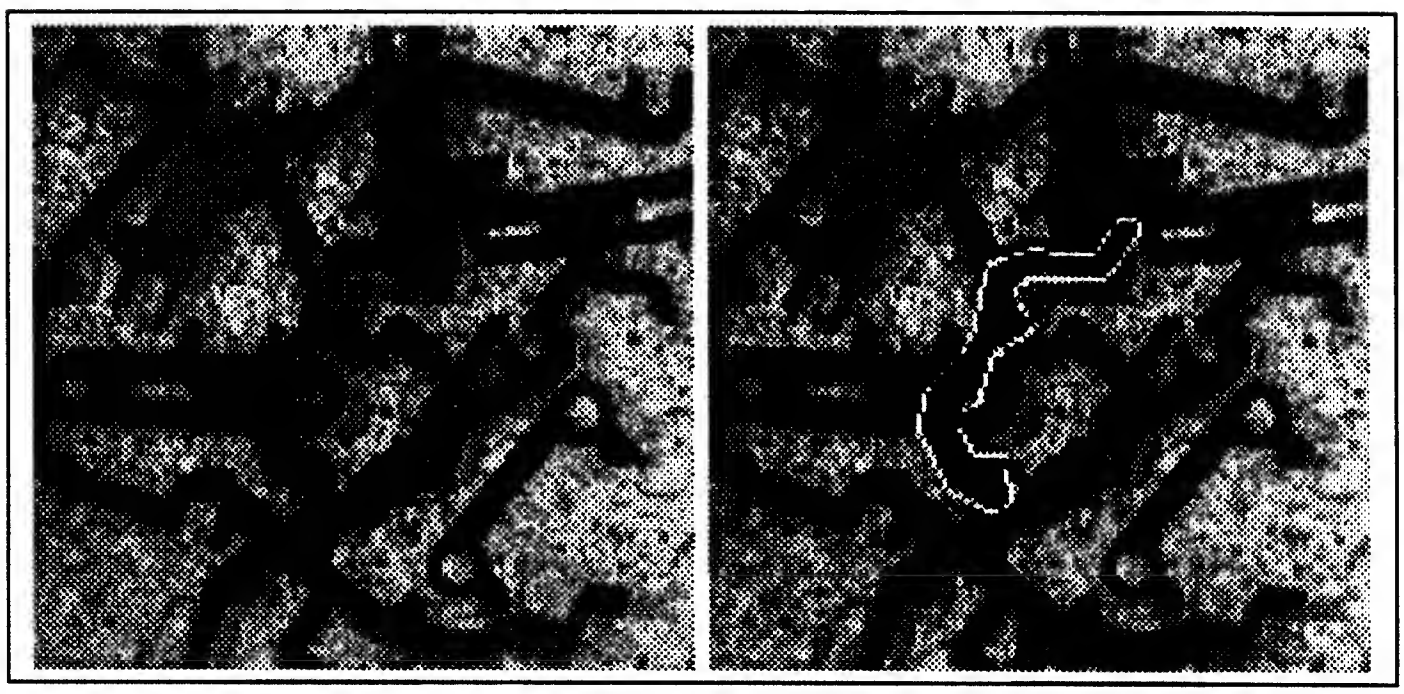

Figure 5.24: The image data, a correct hypothesis, and the image features.

δt_{pix} \ $\delta\phi_{deg}$	2	3	4	5	6	7	8	9	10	11	12
2	1	1	1	1	1	1	1	1	1	1	1
3	1	1	1	1	1	1	1	1	1	1	1
4	1	1	1	1	1	1	1	1	1	1	1
5	1	1	1	1	1	1	1	1	2	1	1
6	1	1	1	1	1	1	1	1	*	1	1
7	1	1	1	1	1	1	1	2	1	*	1
8	2	2	2	1	1	1	1	2	1	1	1
9	1	1	1	1	1	1	1	1	1	1	1
10	1	1	1	1	1	1	1	1	*	1	1

δt_{pix} \ $\delta\phi_{deg}$	2	3	4	5	6	7	8	9	10	11	12
2	0.28	0.29	0.28	0.26	0.28	0.22	0.25	0.29	0.23	0.25	0.28
3	0.28	0.29	0.28	0.26	0.28	0.22	0.23	0.29	0.23	0.23	0.28
4	0.28	0.28	0.28	0.26	0.28	0.22	0.23	0.27	0.20	0.23	0.28
5	0.28	0.29	0.28	0.26	0.28	0.22	0.25	0.29	0.18	0.20	0.28
6	0.28	0.29	0.28	0.26	0.28	0.22	0.18	0.29	*	0.21	0.28
7	0.28	0.28	0.28	0.26	0.28	0.22	0.17	0.20	0.17	*	0.28
8	0.23	0.20	0.23	0.20	0.23	0.17	0.20	0.20	0.20	0.20	0.23
9	0.23	0.23	0.23	0.20	0.23	0.17	0.20	0.19	0.20	0.20	0.23
10	0.23	0.23	0.23	0.20	0.23	0.17	0.19	0.22	*	0.19	0.23

δt_{pix} \ $\delta\phi_{deg}$	2	3	4	5	6	7	8	9	10	11	12
2	193.09	129.49	95.44	77.27	64.96	31.7	31.8	31.93	31.93	31.98	31.85
3	85.32	57.24	42.18	34.14	28.74	14.02	14.08	14.13	14.1	14.12	14.1
4	48.25	32.34	23.82	19.29	16.23	7.9	7.94	7.97	7.96	8.0	7.94
5	29.91	20.07	14.78	11.98	10.08	4.9	4.92	4.96	4.96	4.95	4.94
6	21.31	14.29	10.54	8.52	7.17	3.49	3.52	3.54	*	3.54	3.52
7	15.75	10.56	7.77	6.33	5.29	2.59	2.6	2.61	2.61	*	2.59
8	12.04	8.07	5.95	4.82	4.04	1.98	1.98	2.0	1.99	2.01	1.97
9	9.57	6.42	4.74	3.83	3.21	1.57	1.58	1.59	1.59	1.59	1.57
10	7.76	5.20	3.83	3.11	2.61	1.27	1.27	1.28	*	1.28	1.27

Figure 5.25: Top: The rank, among qualitatively different hypotheses, of the transformation sample point judged correct. Middle: For the transformation sample point judged correct, the fraction of the model perimeter explained by the image features for matches feasible at this sample point. Bottom: For the transformation sample point judged correct, the average number, K_{ave}, of transformation sample points considered per feature match. $m = 27$, $n = 270$, $D = 5$ pixels, and $\Theta = 6$ degrees. The *'s indicate that the correct hypothesis was not in the top 10 qualitatively different hypotheses, or accounted for less that 10% of the model perimeter. The image is shown in figure 5.24.

Figure 5.26: The modeled objects.

5.4 Analysis and Discussion

5.4.1 Uniform Sampling

Uniform transformation sampling was remarkably effective in hypothesizing the correct feature matching and transformation. The results of simulation of the technique on synthetic data are shown in figure 5.4. With sampling intervals of 1 pixel and 1 degrees an average of 95% of the features were *correctly* matched. As the sampling intervals were increased, the technique displayed very graceful degradation in effectiveness. For over half of the sampling intervals shown, more that 50% of the model features were correctly matched. Figure 5.5 displays the result of the same experiment when a total of 164 spurious data points were included, in realistic structural configurations, to make a total of 220 data features. The results are in close agreement with those of figure 5.4, indicating that spurious data have very little effect on the performance of the technique. This is because only matchings that are mutually consistent within the uncertainty bounds at some transformation are hypothesized. Because the correct matchings and transformations due to the model are usually found, it is unlikely that an accidental alignment will produce as large a feasible matching as the correct matching. This breaks down under heavy occlusion when a smaller fraction of the model is visible. In this case, verification is necessary to distinguish between similarly ranked incorrect and correct hypotheses.

Several experiments were conducted on real images, using 3 different objects in several different images. In all the scenes the object of interest was extremely heavily occluded, however, uniform sampling was very effective at computing a correct hypothesis. Typical examples are shown in figures 5.10 and 5.12. In these examples, the transformation sample point with the maximal value of $F(T)$, which in this case is the sum of the image feature perimeter in the largest feasible matching, was the correct hypothesis. In some cases, even at sampling intervals as coarse as every 12 degrees and 10 pixels, the highest ranked hypothesis was the correct one. In all the cases where the occlusion was not too severe, the optimal transformation bucket was in fact the correct hypothesis. Thus this technique can function as an entire recognition engine in cases where occlusion is not too severe. If we are willing to set a threshold on the amount of perimeter that must be accounted for by a hypothesis in order to accept it as a verified recognition result, then the experiments suggest that uniform sampling works extremely well as an entire recognition engine when reasonably fine sampling intervals are used.

In the case where there is severe occlusion, the highest ranked hypothesis is not the correct one, but rather it is typically in the top 10 or 20 hypotheses. It is these cases where verification is necessary, because the best hypothesis cannot be taken as the recognition result. Even so, in cases of extreme occlusion, the correct hypotheses was among one of 10 or 20 hypotheses, and combined with a verification step, uniform sampling provides an hypothesis generator that is extremely robust under occlusion.

It is interesting to consider the times when the correct hypothesis was not the top, or near the top of the hypotheses. Even in successful cases where the top ranked hypothesis was correct such as figure 5.10, inspection of the image seems to indicate that much more than the 33% of the contour found in the best hypothesis is actually visible. This indicates that the feature extraction procedure is failing to provide adequate features for all the visible contour, accounting for the poor performance in heavily occluded cases.

A notable failure of the system to hypothesize correctly is shown in 5.14. Close inspection of the image shows that in fact, very little of the contour is actually visible, and inspection of 5.14 shows that very few features were available for recognition.

5.4.2 Probabilistic Sampling

It is very interesting to compare the performance of the probabilistic sampling technique to that of the simple alignment technique. The results of the experiment indicate that alignment is reasonably effective at determining a reasonable sized matching. Here, alignment

is relying on the distribution of measurement error in position and orientation to find a correct transformation. There is some probability that at least one of the image features will be close enough to the correct pose that the nominal transformation computed from it falls in the match region for many of the correct matches. The interesting thing about the probabilistic approach is that the quality of the transformation found can be improved arbitrarily by considering more probabilistic samples. In the experiments, the effectiveness of the alignment technique compared to the probabilistic technique was similar for $k = 1$, but probabilistic sampling was much better as k was increased. Thus probabilistic sampling accounts more carefully for uncertainty in pose measurement, rather than relying in a simple fashion on the uncertainty distribution.

Chapter 6

Related Work

6.1 Model Based Recognition

The fundamental idea behind model-based object recognition is that objects can be recognized and distinguished based on attributes that capture the information relevant to this task. These attributes, or features, form an abstract *model* of the object. Key to the approach is the idea that it is possible to determine the correspondences between features represented in the model and features abstracted from input sensory data.

There are many attributes of an object which might be exploited for recognition such as color, texture, characteristic motion, and shape. Most of the recognition work in the field of machine vision has focused on recognition by object shape, and this is the focus of this thesis. The approaches to model-based recognition based on shape can be grouped into three broad classes:

1. Techniques which abstract almost completely away from the spatial structure of the object and exploit features of this nature.

2. Techniques which maintain qualitative abstractions of the spatial structure of the object to facilitate recognition.

3. Techniques which maintain explicit quantitative representations of the spatial structure of the object to facilitate recognition.

Techniques belonging to the first group above fall in the domain of classic pattern recognition based on feature spaces. Examples of these types of recognition techniques include the work in recognizing isolated objects from binary images[20] based on *global*

features such as area, perimeter, Euler numbers, and moments of inertia. The use of extended Gaussian images for object recognition is another example among systems dealing with 3D objects. In these types of systems the abstraction process does not maintain information on the spatial structure of the object represented, and an infinite number of qualitatively different objects map to the same representation as abstract features. The features forming the object model are *global* in the sense that changes in the shape of the object in a local region have an effect on the entire object representation.

The second class above contains those systems which maintain a qualitative representation of the spatial structure of the object. In these techniques, features take the form of primitive structural elements, possibly of only local spatial extent, such as descriptions of boundary curve segments. The spatial structure of the objects is maintained in the form of qualitative spatial relations such as **adjacent**, **above**, and **symmetric**. These spatial relations can be represented in the form of relational graphs, in which case feature correspondences are determined through graph matching techniques[11]. The primitive elements may also represent symbols in a formal grammar, in which case feature correspondences are determined by syntactic analysis of feature strings.

Most of the demonstrably effective techniques, robust under object occlusion and in the presence of unknown *spurious* objects, belong to the third category. These techniques maintain in the model and exploit explicit quantitative representations of the spatial structure of the object. Most popular among these is the subclass of techniques dealing with rigid objects and models, in which the geometrical structure is exploited in determining correspondences between the model and the input data.

The recognition technique described here based on transformation sampling belongs to the class of techniques exploiting an explicit geometrical representation of rigid objects. This chapter compares the present work to other studies involving techniques of this type.

6.2 Rigid Geometric Representation Based Techniques

A useful definition of object recognition in robotics includes both pure recognition, that is, determining the presence and identity of an object, and the determination of its position and orientation for manipulation. When rigid geometric models are used to represent objects, the solution to these two tasks complement one another. Again, the fundamental assumption is that it is possible to determine a correspondence between model features and features abstracted from the input data. A suitable correspondence achieves the task

of pure recognition and, in the case of rigid geometric objects and models, facilitates the determination of object pose.

The existing techniques vary in their methods for accomplishing these tasks. A common paradigm for recognition in this class is *hypothesize and test*, in which an hypothesis as to the identity and pose of an object in the environment is constructed, followed by a careful verification of the hypothesis. To some degree almost all systems in this class can be cast in this form. Clemens[13] explored the tradeoffs between various formulations of this basic paradigm.

For this class of recognition techniques, both the model and the sensory data are represented in the same manner. Features are generally primitive elements representing local sections of the surface or boundary contour of the object. The basic idea is that the transformation between the pose of a model feature and the pose of a data feature is approximately the same for all corresponding model and data features. This fact is exploited to accomplish recognition using a variety of strategies. Common to all approaches is the construction of a feature *matching* or *mapping* identifying mutually consistent feature matches. Because the object is rigid, a matching defines an hypothesis as to the object's identity as well as a range of transformations indicating the object's pose. The method for constructing the feature matching is the main difference between different systems.

6.2.1 Hough Transform Methods

Clustering and variations of the Hough transform applied to object recognition have been studied as recognition techniques[32][31][13][26][14][23]. The basic idea of this variation of the Hough transform is that, in the two dimensional case with known scale for example, any pair of model and data features implies a relative transformation which will align their poses, so each match defines a point in **TPS**. Call these *match-points*. Since the model is rigid, correctly matched features will yield approximately the same relative transformation, when rotation precedes translation, and thus a cluster of match-points will be formed in parameter space. Good candidates for the correct transformation are found by searching parameter space for such a cluster. In principle, this is an excellent approach. In the ideal case there would be a sharp peak in parameter space corresponding to the correct transformation. In practice, there are two main factors which make identifying the correct transformation difficult: error in the measurement of the pose of image features, and spurious points in parameter space due to incorrect matches.

There is a strong coupling between the rotation component and the translation com-

ponent of the transformation aligning a model feature with an image feature. An error in the estimate of the orientation of an image feature results in an incorrect relative rotation, which in turn results in an incorrect relative translation. This has the effect of spreading out the cluster in space, so that although features are correctly matched, they do not all correspond to the same point in parameter space. Thus pose uncertainty causes dilution of clusters making finding the clusters difficult.

The match-points in parameter space due to incorrectly matched features contribute to what might be called the overall *background noise* of the parameter space. As the level of this background noise increases, that is, as the number of points in parameter space due to incorrect matches increases, peaks or clusters in parameter space become harder to distinguish.

These clustering techniques are commonly implemented by tessellating parameter space into cells or *bins*. The bins with many points in them determine clusters. The simplest approach is to simply divide the space into cells, compute the nominal transformation aligning each of the feature matches, and place a vote in the bin containing that transformation. Choosing the bin size is difficult for the following reasons. Because the cluster of match-points from correctly matched features will be spread out due to pose error, the bins must be made large enough to include all of a cluster. There is still the problem that a cluster may be split over several buckets. One solution to this problem is to overlap the buckets. The main problem with large buckets, however, is that they integrate the space of transformations over a large region, and accumulate contributions from matches that may not be mutually feasible at any transformation. The bin with the most matches falling in it may not be associated with the optimal transformation because it is quite possible that no single transformation will simultaneously align all, or even most, matches in the bin. We can see this by noting that the regions of feasible transformations for two different matches may intersect the same bin, but not intersect one another, creating a false peak in the tessalated parameter space. See figure 6.1.

Simply making the bucket size smaller will not suffice, because in this case any one bin may only contain a fraction of the cluster of match-points from correct matches, and thus the peak may not be detected. The discussion on the structure of parameter space earlier suggests an approach that will work properly. First, make the bin size small enough so that it is likely that most match-points in them are feasible at a common transformation. Second, consider a given feature match at all the possible transformations possible under the effects of pose measurement error, and place a vote in all possible buckets. In the framework of chapter 3, place a vote in all the buckets intersected by the match region[13]. Unless

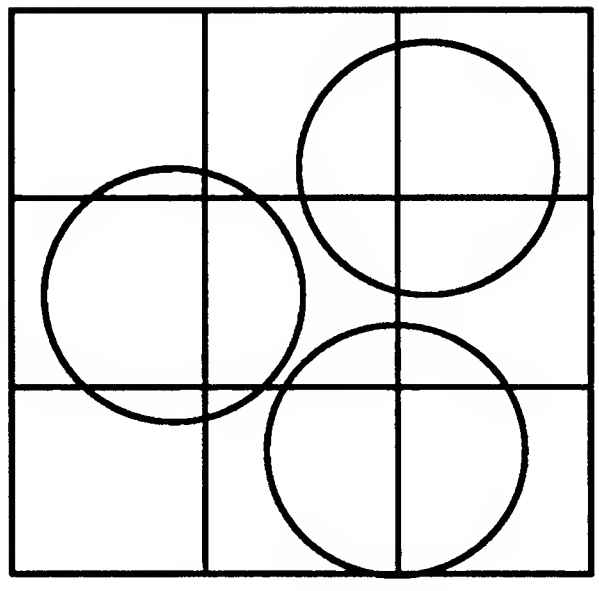

Figure 6.1: The squares represent slices of Hough bins, the circles represent regions of valid transformations for matches. Although a Hough bin may have several members, they are not necessarily mutually consistent.

great care is taken in filling the bins, the clustering technique alone is not adequate for recognition, although it has been used as a coarse filter of potential matches before a more detailed procedure. For example, Grimson and Lozano-Pérez[16] simply use the Hough transform to roughly order the matches they will consider in a subsequent constraint-based search procedure.

6.2.2 Transformation Sampling is not a Hough Transform

The idea of uniform transformation sampling was first introduced in response to the difficulties involved with correctly implementing the Hough transform. Both transformation sampling and Hough transform are techniques set out to accomplish the same task, identify transformations which are feasible for a large number of feature matches. Uniform transformation sampling in particular is very close to the idea of a Hough transform, although there are some important differences.

The primary difference between Hough transform clustering techniques and uniform transformation sampling is that in Hough transform methods an integration of information is performed over finite sized regions of **TPS**, while in uniform sampling, information is taken only at a point. From one viewpoint, uniform sampling can be related to the Hough transform by considering the size of the cells to be infinitesimal, and the spacing of the

cells in each dimension to be equal to the the transformation sampling intervals. The distinction between finite regions and points in **TPS** is an important one, only true peaks are found with uniform sampling, corresponding to matchings which satisfy the constraints on uncertainty at a given transformation. With the Hough transform, depending on the size of the bucket, false peaks can exist where no globally consistent matching exists for any transformation.

While in principle the Hough transform can account for pose uncertainty in the same way done in this work, by carefully considering match-regions, most existing implementations do not. In contrast, transformation sampling explicitly and accurately accounts for pose uncertainty due to sensing errors and feature fragmentation, and accounts for any amount of uncertainty.

False Positives versus False Negatives

We can characterize the main differences between uniform transformation sampling and Hough transforms by noting that transformation sampling in a sense trades false negatives for false positives. We have argued that the Hough transform may indicate a popular transformation which in fact is not globally consistent. On the other hand, uniform transformation sampling will never indicate anything but a globally consistent transformation. Because we are sampling parameter space, however, it is possible that we could miss a peak in parameter space, and thus miss a transformation which the Hough transform may have found. It was shown in 3 that under assumptions as to the nature of measurement errors, it is possible to sample within the optimal region of **TPS** with high probability. The possibility of missing the optimal region is only true for uniform transformation sampling and probabilistic transformation sampling. Critical point sampling will always find the optimal regions of parameter space.

One of the techniques studied by Clemens[13] has many similarities to the two probabilistic techniques developed here. He associates with each point in parameter space a probability density characterizing the probability that, for a given matching at the particular transformation, the observed image would have occurred. This is based on the conjunction of the probabilities of the observed difference in pose between matched features after transformation. In principle the probability should be computed for all transformations and all matchings, but he makes use of the bounded uncertainty assumption to exclude matches from matchings in which the two features do not align well enough after transformation. He noted that due to errors, the region of possible transformations for a given match is a

helical cloud, and his technique filled all buckets intersecting this cloud with a pointer to the match and an indication of the probability of that particular transform.

6.2.3 Constraint Satisfaction

Many approaches make the assumption that the error or uncertainty in feature pose measurement is bounded. This provides great constraint on what matchings are possible. One approach to hypothesis generation is to construct maximal sets of mutually consistent feature matches under the uncertainty constraints. This provides a matching and possibly a range of transformations which can then be verified. Examples of this are RAF due to Grimson and Lozano-Pérez[16], LFF due to Bolles and Cain[8], the approach by Baird[4], and the system by Koch and Kashyap[24]. All these systems build sets of pairwise consistent feature matches to compute very careful hypotheses as to the identity and pose of an object from the sensed data, which are then carefully verified. These might be viewed as constraint satisfaction techniques. All of these systems make the assumption that the uncertainty in feature pose is bounded, and thus depending on a given range of transformations, certain matches are impossible. The constraints applied by RAF, LFF, and Koch and Kashyap are pairwise consistency constraints, while those by Baird are global consistency constraints.

The goal of the RAF system is to find a matching that is feasible, in the sense of global consistency, that has a maximal value of a quality measure, which in their case is the amount of model perimeter accounted for by the matching. The tree search procedure which builds matches only relies on pairwise consistency constraints, however, in constructing matchings, although Grimson and Lozano-Pérez found that these constraints capture most of the constraint provided by the global consistency constraint. Search limiting heuristics are employed to reduce the system's tendency to thrash making small changes to essentially similar matchings.

The transformation sampling approach and RAF have much in common in the assumptions and goals. The main difference is the technique for constructing optimal matchings. Critical point sampling is provably correct under the same assumptions and goals and has polynomial complexity in terms of features. RAF has exponential expected complexity in terms of model and image features in the case with spurious image features, but expected polynomial complexity in the case of isolated objects[15]. RAF may not find the optimal solution, however, due to search limiting heuristics. One important difference between critical point sampling and RAF is that the latter utilizes extended features, as well as point

features, while the extension of the former to this case has not yet been investigated.

While transformation sampling is highly parallel in nature, it appears the sequential nature of the constrained search is what makes it effective, and parallel implementation without exploding a large part of the search space may not offer substantial speedup in parallel. Finally, it must be noted that RAF does work with 3D data and 3D objects. While it appears that transformation sampling will extend to 3D, this must be investigated further.

The LFF approach and the approach of Koch and Kashyap are similar to RAF in that they utilize pairwise constraints based on uncertainty bounds to build matchings. These three systems solve constraint satisfaction, or consistent labeling problems to build consistent matchings, however consistency constraints are pairwise rather than global. Transformation sampling is also solving a constraint satisfaction or consistent labeling problem to determine matchings. In this case however the consistency constraints exploited are global in nature rather than simply pairwise.

The system by Baird formalizes the matching problem as a constraint satisfaction problem using a linear programming approach. He shows that in expected $O(n^2)$ time, where n is the number of features, all feasible (consistent with the constraints) matchings can be found. An important and extremely limiting restriction he applies is that there are no spurious or missing data. In contrast, critical point sampling handles spurious and missing data, and offers an optimal solution in polynomial time, under the given formulation of the goals and measures of optimality.

6.2.4 Alignment Techniques

There is a class of recognition techniques which makes a different tradeoff between the hypothesis generation and verification steps, placing less emphasis on the generation of careful hypotheses and more emphasis on the verification step. Ullman has called these techniques recognition by *alignment*[33][22]. Other examples of alignment related techniques are found in [3][13][28]. Rather than build a complete matching between model and data features as transformation sampling and the two systems above do, the idea is to determine a partial matching which is sufficient to solve for a transformation. Because the model is rigid, this transformation serves to construct a hypothesis of the object's pose. Again assuming bounded uncertainty in data feature pose measurement, after transformation the hypothesis can be strengthened by finding more feature matches, where matches are only possible between features that are sufficiently close, measured as a function of the

uncertainty bounds. When the strength of a hypothesis is sufficient it can be passed to a verification procedure.

In 2D, with known scale, only a single match between oriented features is required for alignment. If scale is unknown, a pair of matches is required. For 3D objects using 3D features, a pair of matches is require. For 3D objects from 2D features, Huttenlocher and Ullman[22] have shown that only three feature points are required for orthographic projection and scaling. For the 2D case with oriented features, known scale, and m and n model and data features respectively, all possible feature matches provide $O(mn)$ hypothesis. For each hypothesis we must in the most naive method do $O(mn)$ work to check for support of the hypothesis. If the data features are organized in some sort of spatial hash table, then only $O(m)$ work need be done for each hypothesis in typical cases, resulting in an overall complexity of $O(m^2n)$. Huttenlocher has shown that in the 3D from 2D case, hypothesis generation and evaluation requires $O(m^3n^2)$ when the data features are spatially hashed. Thus the complexity of alignment approaches is quite favorable. Because many of the operations performed are on 2 or 3 features and are highly local, much of the computations required appear to be naturally parallel, although no parallel implementation has been demonstrated.

In one sense, alignment and transformation sampling are very similar. Both take transformations as hypotheses and then verify them for recognition. Alignment takes the *nominal* transformation which aligns the measured pose of a small number of feature matches as the hypothesis. The transformation sampling techniques allow for the fact that the correct transformation could fall anywhere in the match region. Thus, one difference between the transformation sampling approach and the alignment approach is that the former explicitly accounts for error in feature position, while the latter does not. With alignment, even if an initial hypothesis is composed of correctly matched features, if the error in some of the data features is great enough, the nominal transformation could be far enough off that the hypothesis will be incorrectly rejected since other features do not align adequately. This is especially true in the case of fragmented features, and matching point image features to extended model features, where the nominal transformation may be considerably different from the correct transformation.

Nonetheless, the same type of probabilistic analysis applied to transformation sampling can be applied to alignment. If the error in the measurement of image feature pose is characterized by a probability distribution, it should be easy to characterize the probability that one of the nominal transformations derived from correct feature matches is close enough to the correct transformation that it falls in the match region of many other correct

matches, and applying this transformation adequately aligns the model with the image. In this view, in the limit as the number of visible model features in the image goes to infinity, the probability that the correct transformation is not found goes to zero; in the limit alignment will always work. Alignment implicitly relies on the probability distribution of image feature pose error. From this point of view, probabilistic sampling and alignment are very close, however probabilistic sampling allows the probability that the correct hypothesis is found to be increased arbitrarily.

Chapter 7

Extensions and Future Work

7.1 Three-Dimensional Object Recognition

Of interest in the study of the techniques for 2D recognition described here, is their extensibility to the case of 3D object recognition from 3D or 2D sensory data. It appears that the techniques of uniform sampling and probabilistic sampling can be extended to the case of 3D objects using 3D data. The extension of the critical point analysis to the 3D case, however, is not as clear.

3D Transformation and Uncertainty

Analogous to the case of 2D model and image features, we can define 3D features to have 3 positional, and 3 orientational degrees of freedom. Of fundamental importance in the case of 2D recognition is that a feature match defines a unique transformation which aligns the poses of the two features. This is not true for 3D features, as can easily be seen by considering the unit vectors representing the orientations in 3-space of a model and image feature. The rotation about some origin which aligns the model feature's orientation with that of the image feature is parameterized by 2 angles, but any third rotation about an axis parallel to the image feature leaves their relative orientation unchanged. A pair of feature matches, however, provides enough constraint to derive a unique rotation.

Define rotation of vectors in $\Re^3$ by rotation of θ about an axis defined by a unit vector $\hat{\omega}$. This formulation of rotation in three dimensions is easily represented by unit quaternions $\mathbf{r} = r_0 + r_x\hat{i} + r_y\hat{j}r_z\hat{k} = \cos(\frac{\theta}{2}) + \sin(\frac{\theta}{2})\hat{\omega}$, see for example [21]. Using quaternion product, the result of rotating a vector $\vec{v}$ is given by $\vec{v}' = \mathbf{r}\vec{v}\mathbf{r}^*$, where $\mathbf{r}^*$ is the conjugate of the quaternion $\mathbf{r}$, formed by negating $\hat{\omega}$.

Consider two model features and another pair of features derived by rotating and translating these, which can be considered image features. As can easily be shown, any two vectors v_1, $v_2 \in \Re^3$ can be aligned in orientation by rotating by an angle θ about an axis $\hat{\omega}$ lying in the plane bisecting the angle between the two vectors. The unique (up to sign) vector $\hat{\omega}$ about which both model features are rotated by some angle θ to align their orientations is given by intersecting two such planes due to the two different pairs of model and image orientation vectors. Let $\hat{n}_m$ and $\hat{n}_d$ represent the orientation of the normal vectors describing the orientation of a model and image feature respectively. Then from the above argument it is easily shown that

$$\hat{\omega} = \frac{(\hat{n}_{m_1} - \hat{n}_{d_1}) \times (\hat{n}_{m_2} - \hat{n}_{d_2})}{|(\hat{n}_{m_1} - \hat{n}_{d_1}) \times (\hat{n}_{m_2} - \hat{n}_{d_2})|}$$

Given $\hat{\omega}$, $\hat{n}_{m_1}$, and $\hat{n}_{d_1}$, the rotation θ can then be determined, see for example [17].

Thus, given a pair of 3D feature matches, where one pair is a rotated, translated version of the other, it is possible to solve for the quaternion representing the rotation which will align their orientations. The translation can then simply be solved for by the difference in the feature positions. When error in feature extraction is considered, however, this becomes more difficult. Assume that we are dealing with 3D point features. Let the correct orientation of an image feature be given by $\hat{n}_{d_0}$ and its measured orientation by $\hat{n}_d$. To bound the error, assume that $\hat{n}_{d_0} \cdot \hat{n}_d \geq \cos(\Theta)$, thus the measured orientation $\hat{n}_d$ lies in a cone centered at the correct orientation $\hat{n}_{d_0}$ or equivalently the correct orientation $\hat{n}_{d_0}$ lies in a cone centered at the measured orientation $\hat{n}_d$. This error model was used by Grimson and Lozano-Pérez[16]. Let the correct and measured positions of the image feature be given by $\vec{p}_{m_0}$ and $\vec{p}_m$, respectively. Assume that $|\vec{p}_{m_0} - \vec{p}_m| \leq D$. Thus the measured position of the features falls in a sphere centered at the correct position.

To account for the uncertainty in orientation, any rotation satisfying $|\hat{n}_d \cdot \mathbf{r}\hat{n}_m\mathbf{r}^*| \geq \cos(\Theta)$ is possible. Thus there is a set of vectors $\hat{\omega}$ which are possible, and for each vector $\hat{\omega}$ there is a range of rotations θ possible. Then, for any feasible rotation there is a range of translations $\mathbf{T}$, after rotation, such that $|\vec{p}_d - \mathbf{T}\mathbf{r}\vec{p}_m\mathbf{r}^*| \leq D$.

As in the case with 2D recognition, the range of feasible transformations for a given pair of matches corresponds to a region in a 6 dimensional parameter space. The intersection of these match-regions are 6 dimensional intersection-volumes analogous to those defined in chapter 3. We can now consider the idea of hypothesizing a feature matching by searching parameter space for intersection volumes consistent with large numbers of pairs of feature matches.

Uniform Sampling

For any pair of feature matches, the set of feasible rotation vectors $\vec{\omega}$ corresponds to a region on the unit sphere. This region can be sampled at points on the sphere. For each sample vector $\vec{\omega}$, the range of feasible rotations θ about $\vec{\omega}$ can also be sampled. Finally, for each complete rotation sample, after rotation, the range of possible translation can be sampled on a 3D grid. Brou explored related issues involved with sampling the space of 3D orientations in matching object Extended Gaussian Images[9].

Following the construction of transformation sample points, the algorithm for hypothesis generation is exactly similar to that of the 2D case. Chapter 5 has shown in the 2D case that the number of transformation sample points that need to be considered per feature match is reasonable. In the 3D case, however, the complexity increases considerably. First, instead of considering K_{2D} sample points for each of mn feature matches, in 3D we must consider K_{3D} sample points for each of m^2n^2 pairs of feature matches. Second, because there are 6 parameters instead of 3, it is likely that K_{3D} is considerably larger than K_{2D}. If, however, as in the 2D case, reasonable matching hypotheses can be constructed by sampling quite coarsely, this approach may be feasible in practice. The asymptotic complexity in terms of the input feature sets is quite favorable, of $O(m^2n^2)$.

Probabilistic Sampling

The probabilistic sampling approach should extend easily to the 3D case. Generating a random point in the 6 dimensional match-region is easily done by perturbing the parameters of the nominal transformation within the uncertainty bounds. The important thing to be determined is how to sample points according to the appropriate probability distribution. The attractive aspect of probabilistic sampling in 3D is that it is much simpler than even the uniform sampling technique.

Critical Point Sampling

Unfortunately, in 3D there are no critical points falling on the boundaries of intersection-volumes as there were in the 2D case. Their analogues are likely to be hypersurfaces. Nonetheless, the idea of not attempting characterization of the whole intersection-volume, but rather places in the space where the character of the space changes at their boundaries, is a very intriguing research direction.

7.2 Hypothesis Refinement and Verification

The recognition strategy formalized in chapter 3 is composed of hypothesis generation, refinement, and verification. In the experiments, however, the system only utilized the hypotheses generation step by transformation sampling. In fact the hypotheses were so good that no further refinement or verification was performed.

In the case of many similar objects, simple objects, or heavy occlusion, the hypothesis generation step alone will not suffice for the entire recognition engine, and hypothesis refinement and verification will be required, although the transformation will still generate relatively few high quality hypotheses. Further investigation into methods of hypothesis refinement and parallel verification is needed.

7.3 Critical Point Sampling

There are a number of obvious refinements of the 2D critical point sampling method that would increase its flexibility and power. First, it should be possible to deal with unknown scale by introducing a scale parameter s in the transformation with the rotation: $se^{i\phi}$. The same ideas of search for critical points may still work. Second, the analysis of the behavior of match-circles in determination of the location of critical points should apply in a similar fashion to match-region with non-circular cross section. In particular, the use of line segments as features would lead to match regions with approximately rectangular cross sections. It would be interesting to apply the same analysis in these cases.

7.4 Other Extensions

Large Model Libraries

Any recognition technique is extendable to large libraries at a cost linear in the size of the object library. The techniques studied here do not extend to the case of large libraries, unless the models are considered in this way, serially. It is possible, however, to consider a small number of different models simultaneously. Several such computations could take place in parallel if the computations were kept orthogonal, by labeling features due to different models as such, and treating them completely separately from one another.

Multiple Instances

All of the techniques are directly extendable to include any number of multiple instances of the same model. In fact, all the work done to find one instance finds all instances. All that is necessary to do is to pick all optimal hypotheses which pass verification.

7.5 Special Hardware Implementations

It seems that parallel computation is necessary for the fast computation of recognition. Investigating parallel algorithms and implementing them on existing general purpose parallel machines is the first step in understanding the issues involved. It is likely that even greater performance can be achieved through special hardware implementations, and so special hardware implementations of the algorithms are intriguing.

The three transformation sampling techniques described here are simple and highly parallel in nature. For these reasons a special hardware implementation of the algorithms may be possible, allowing very fast recognition times. In particular, probabilistic sampling simply requires the generation of a random point inside a match region, and the related containment computation. These are simple, local operations. The accumulation and ranking of the best sample points, however, is a more global operation, and the requirements of special hardware for this is not clear.

Chapter 8

Conclusions

8.1 Summary

Recognition was defined as hypothesizing a feature matching $\mathbf{M}$, and a transformation T that optimizes a metric $F(\mathbf{M}, T)$ over all possible $\mathbf{M}$ and T, followed by verification of these hypotheses. To make this tractable, the assumption of bounded image feature pose uncertainty was introduced, requiring that, after some transformation T, the pose of the the model and image feature must be less than the bounds of pose uncertainty in position and orientation.

By intersecting the constraints on feasible matches for all possible matches, sets of feasible matches are constructed based on regions of transformation parameter space, **TPS**. Hypotheses of feature matchings can be constructed by defining a particular metric $F(T)$ which facilitates evaluation of these regions of consistent matchings, and then searching **TPS** for optimal values of this metric.

Once found these matchings can be refined to determine the transformation optimally aligning the matched features. The transformation and matching thus hypothesized is then verified using whatever data appropriate to the verification method, such as the model and image data, extracted intensity edges, or the original brightness images.

The technique of transformation sampling was introduced as a method for searching **TPS** for optimal values of $F(T)$. Three methods of transformation sampling were discussed: uniform sampling, probabilistic sampling, and critical point sampling.

8.2 Conclusions

The critical point sampling analysis provides an elegant and simple solution to the particular formulation of the matching problem where uncertainty has fixed known bounds and the goal is to find maximal feasible matchings. In past work, computationally complex methods involving graph search and tree search have been used for this problem, and while they are quite effective in practice, have unacceptable worst case bounds. In contrast, the complexity of the critical point sampling technique is polynomial in the size of the inputs, and provably correct. Furthermore, the critical point sampling technique is highly parallel in nature and particularly well suited to parallel implementation. At this point, however, the critical point sampling analysis serves only as a theoretical discussion of the complexity of this formulation, and has not been implemented.

The two sub-optimal, but simple and highly parallel techniques of uniform and probabilistic transformation sampling are approximate methods to finding regions of **TPS** defining large feasible matchings. Experiments indicate that these techniques are effective in finding correct matching and transformation hypotheses. In particular, the uniform sampling technique produces correct hypotheses sufficient for recognition even with coarse sampling intervals. Thus the computational resources required are reasonable, and the algorithm is practically realizable on existing parallel machines.

Appendix A

The expand-vector, outer-product, and generalized-histogram operations illustrate some interesting and powerful parallel operations. It is interesting to consider their implementations in terms of elementwise, scan, permute, and sort primitive operations.

Expand Vector

The **expand-vector**(A,K) function takes a vector A of data objects, and an equal sized vector of integers K, and returns a new vector where each element $A[i]$ is represented in $K[i]$ contiguous vector elements of the result. The new vector has length $\sum_i K[i]$. The complimentary function **expand-vector-index** returns a vector of integers assigning a zero-based index to each copy of a particular element $A[i]$ determining its position in the segment of copies. The first step is to form a vector A_1 where each element $A[i]$ of A is represented with $K[i]-1$ empty elements after it. This is done by perfoming a **+-scan** on the vector K to determine the total number vector elements that will be below $A[i]$ in the new expanded vector, then dispersing A by permuting according to this cumulative sum. The $K[i]-1$ empty elements after each permuted element $A[i]$ are filled in by perfoming a segmented **first-scan** on the segments thus defined. The **first-scan** function simply copies the first element in a segment to all the others in a segment. In the following example, A_2 forms the result of **expand-vector**, and Index the result of **expand-vector-index**.

A	=	$[a_0$	a_1	a_2	$a_3]$			
K	=	[2	3	2	1]			
A_1	=	$[a_0$		a_1			a_2		$a_3]$
S	=	[T	F	T	F	F	T	F	T]
A_2 = **first-scan**(A_1,S)	=	$[a_0$	a_0	a_1	a_1	a_1	a_2	a_2	$a_3]$
Index	=	[0	1	0	1	2	0	1	0]

Outer Product

The outer product is analogous to the Cartesian product of two sets $\{< a_i, b_j >\} = \{a_i\} \times \{b_i\}$, taking two vectors as input and returning a vector whose elements consist of all possible pairs of elements in the original two vectors. As an example, if A represents a vector of length m of model features, and B represents a vector of length n of image features, the outer product is a vector of length mn whose elements are all pairs of the elements of A and B. The outer product is implemented using a constant number of permutation and scan operations. The first step is to perform **expand-vector**(A, N), and **expand-vector**(B, M), where N is a vector whose elements are all n, of length equal to A, and M is a vector of length equal to that of B whose elments are all m. This forms two vectors consisting of n copies of each element of A, and m copies of each element of B. The last step is to permute the elements of B to align all pairs of elements at the same index. To do this $B[i]$ is permuted to index $j = n(i \bmod m) + \lfloor \frac{i}{m} \rfloor$ to form B_3.

$$
\begin{array}{lcl}
A & = & [a_0 \quad a_1 \quad a_2] \\
N & = & [4 \quad 4 \quad 4] \\
A_2 = \textbf{expand-vector}(A, N) & = & [a_0 \quad a_0 \quad a_0 \quad a_0 \quad a_1 \quad a_1 \quad a_1 \quad a_1 \quad a_2 \quad a_2 \quad a_2 \quad a_2] \\
B & = & [b_0 \quad b_1 \quad b_2 \quad b_3] \\
M & = & [3 \quad 3 \quad 3 \quad 3] \\
B_2 = \textbf{expand-vector}(B, M) & = & [b_0 \quad b_0 \quad b_0 \quad b_1 \quad b_1 \quad b_1 \quad b_2 \quad b_2 \quad b_2 \quad b_3 \quad b_3 \quad b_3] \\
B_3 = \textbf{permute } B_2 & = & [b_0 \quad b_1 \quad b_2 \quad b_3 \quad b_0 \quad b_1 \quad b_2 \quad b_3 \quad b_0 \quad b_1 \quad b_2 \quad b_3] \\
A_2 & = & [a_0 \quad a_0 \quad a_0 \quad a_0 \quad a_1 \quad a_1 \quad a_1 \quad a_1 \quad a_2 \quad a_2 \quad a_2 \quad a_2]
\end{array}
$$

Generalized Histogram

The histogram maps a vector of keys, K, to a new vector H, where any two elements of K, k_i and k_j, are mapped to the same element of H iff $k_i = k_j$. This equivalence relation defines a partition of the elements of K. Let A be an arbitrary data vector where the elements of A and K are in one-to-one correspondence, thus $|A| = |K|$. The same mapping of K to H also defines a partition of the elements a_i of A. Let $\bar{a}_i$ represent the equivalence class of element a_i. Further, let $\phi(\bar{a}_i)$ be a function $\phi : \{\bar{a}_i\} \longrightarrow \Re$ on the set $\{a_j\} \in \bar{a}_i$. The generalized histogram takes as inputs K and A. The result vector, G, is defined as follows. $G[i] = \phi(\bar{a}_i)$ where $\bar{a}_i$ is the equivalence class of the element a_i corresponding to element k_i of K.

When ϕ is a binary associative operator, the generalized histogram can be implemented efficiently using scans and permutations. First, the vector K is sorted, and equal elements

arranged in contiguous vector elements. The vector A is permuted exactly as K was in sorting, so elements of A and K remain in correspondence. A vector of segment flags is constructed delineating each contiguous segment of equal elements of K. A segmented scan of the elements of the permuted A, using these segments, is computed using scan functions to compute ϕ for each segment with another first scan. The result accumulates in the last element of the segment, and is copied back over the segment. The final result is formed by taking the result of these scans and permuting back to the original ordering of K, completing the generalized histogram.

Bibliography

[1] **Lisp Reference Manual.* Technical Report, Thinking Machines Corporation, Cambridge, MA, 1987.

[2] *Model CM-2 Technical Summary.* Technical Report, Thinking Machines Corporation, Cambridge, Ma., April 1987.

[3] N. Ayache and O. D. Faugeras. Hyper: a new approach for the recognition and positioning of two-dimensional objecs. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 8(1):44–54, 1986.

[4] Henry S. Baird. *Model-Based Image Matching Using Location.* MIT Press, Cambridge, MA, 1985.

[5] P. J. Besl and R. C. Jain. Three-dimensional object recognition. *ACM Computing Surveys*, 17(1):75–145, 1985.

[6] G. Blelloch. *Forthcoming.* PhD thesis, Massachusetts Institute of Technology, 1988.

[7] G. Blelloch. Scans as primitive parallel operations. In *Proceedings Int. Conf. on Parallel Processing*, 1987.

[8] R.C. Bolles and R.A. Cain. Recognizing and locating partially visible objects: the local–feature–focus method. *International Journal of Robotics Research*, 1(3):57–82, 1982.

[9] P. Brou. Using the gaussian image to find the orientation of objects. *International Journal of Robotics Research*, 3(4):89–125, Winter 1984.

[10] John F. Canny. A computational approach to edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 8(6):679–698, 1986.

[11] J. K. Cheng and T. S. Huang. Recognition of curvilinear objects by matching relational structures. In *Proceedings of IEEE Computer Society Conference on Pattern Recognition and Image Processing*, pages 343–348, Las Vegas, June 1982.

[12] R. T. Chin and C. R. Dyer. Model-based recognition in robot vision. *ACM Computing Surveys*, 18(1):67–108, March 1986.

[13] David T. Clemens. *The Recognition of Two-Dimensional Modeled Objects in Images.* Master's thesis, Massachusetts Institute of Technology, 1986.

[14] R. O. Duda and P. E. Hart. Use of the hough transformation to detect lines and curves in pictures. *Communications of the ACM*, 15(1):11–15, 1972.

[15] W. Eric L. Grimson. *The Combinatorics of Object Recognition in Cluttered Environments using Constrained Search.* Technical Report A.I. Memo 1019, Artificial Intelligence Laboratory, Massachusetts Institute of Technology, 1988.

[16] W. Eric L. Grimson and Tomas Lozano-Perez. Localizing overlapping parts by searching the interpretation tree. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 9(4):469–482, July 1987.

[17] W. Eric L. Grimson and Tomas Lozano-Perez. Model-based recognition and localization from sparse range or tactile data. *International Journal of Robotics Research*, 3(3), 1984.

[18] W. D. Hillis and G. L. Steele Jr. Data parallel algorithms. *Communications of the ACM*, 29(12):1170–1183, November 1986.

[19] W. Hoeffding. Probability inequalities for sums of bounded random variables. *Journal of the American Statistical Association*, 58(301):13–30, 1963.

[20] B. K. P. Horn. *Robot Vision.* MIT Press, Cambridge, Mass., 1985.

[21] B.K.P. Horn. Closed-from solution of absolute orientation using unit quaternions. *Journal of the Optical Society of America*, 4:629, April 1987.

[22] D. P. Huttenlocher and S. Ullman. Object recognition using alignment. In *Proceedings of the International Conference on Computer Vision*, pages 102–111, June 1987.

[23] T. N. Mudge J. L. Turney and R. A. Volz. Recognizing partially occluded parts. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 7(4):421–410, 1985.